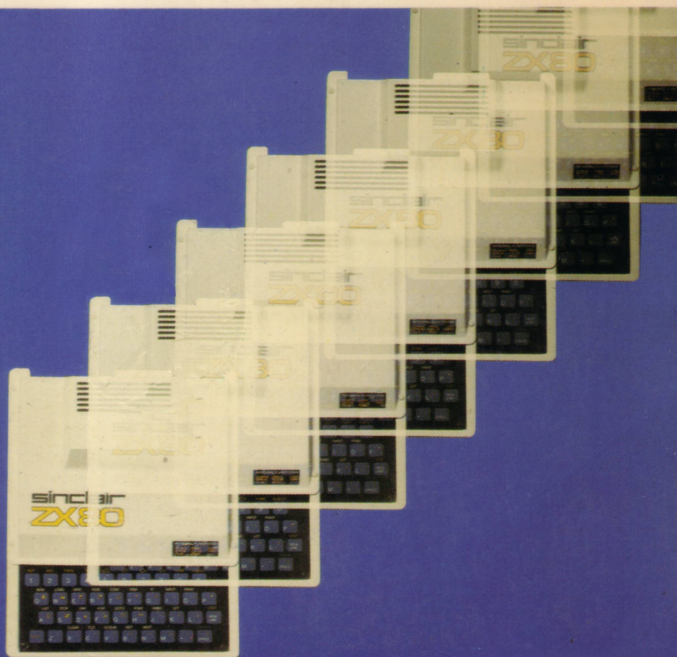
IMPARIAMO A PROGRAMMARE

EDIZIONE ITALIANA

R. BONELLI

GRUPPO EDITORIALE JACKSON ITALIANA

IMPARIAMO A PROGRAMMARE

CON LO ZX80

di
Rita Bonelli

GRUPPO
EDITORIALE
JACKSON
Via Rossellini, 12
20124 Milano

PREFAZIONE

Il Sinclair ZX80 è una delle ormai sempre più diffuse macchinette di costo accessibile ad una vasta cerchia di utenti, tra i quali numerosissimi, anche per questo, sono coloro che si introducono per la prima volta nel mondo affascinante dell'informatica. Grazie ai micro e personal computer, come è stato giustamente affermato da più d'uno, l'elaborazione elettronica dei dati sta perdendo il suo aspetto misterico, diventa "democratica", esce dai confini del sancta sanctorum dei sacerdoti in camice bianco che un tempo erano gli unici depositari di una scienza per soli addetti ai lavori.
Ciò non toglie che, per chi muove i primi incerti e generalmente timidi passi in questa attività, siano tutte rose e fiori. Le difficoltà non vanno certo drammatizzate e tuttavia ci sono: occorre formarsi una mentalità adatta, esercitarsi su esempi opportunamente graduati e significativi in modo soprattutto da non frustrare con insuccessi iniziali tante buone intenzioni e aspirazioni.
Ecco allora libriccini come questo che mirano ad una funzione duplice: non solamente cioè dare le necessarie informazioni sul particolare sistema oggetto del discorso ma chiarire opportunamente quali sono i criteri generali e basilari del suo utilizzo.
Perché, come è noto ed arcinoto (ma, si direbbe, a volte viene dimenticato anche in molti "bug" o persino "cook" books), i sistemi per l'elaborazione dati non si limitano certo al "materiale" (così oggi gli accademici denominano l'hardware, la ferraglia) ma la parte più succosa si riferisce a quel quid di impalpabile che i sapientoni chiamano "logicale", gli inglesi software e che viene quasi la tentazione di indicare con *anima*, dato che senza di essa il corpo non vive.
Forte della sua esperienza didattica l'autrice non si è, giustamente, limitata a tradurre - rendendolo peraltro chiaro in massimo grado - il manuale operativo della Sinclair, ma, partendo da premesse di carattere generale, ha voluto soprattutto porre l'accento sull'uso pratico di un personal computer.
Se si segue la successione dei vari capitoli si può notare come quelli di carattere generale (fondamenti sulla struttura hardware e nozioni basilari di programmazione) si alternino efficacemente a quelli specifici dello ZX80: architettura, set di istruzioni, caratteristiche del suo linguaggio BASIC ecc.
Un altro criterio che ci è parso interessante e valido è quello di anteporre la presentazione del BASIC a quella del linguaggio macchina. Contrariamente cioè a quel che avviene nella maggior parte di testi di questo tipo, nei quali si parte dal presupposto che "per imparare meglio occorre partire dalle basi". Invece, giustamente a nostro avviso, l'autrice ha prima voluto fornire con un linguaggio *orientato al problema* una chiave di accesso all'e.d.p. più potente e semplice al tempo stesso.

Gli esempi svolti sono chiari ed efficaci e, per lo più, hanno un carattere hobbistico: un programma per "giocare a mordere il formaggio" oltre che curioso sposa il sano principio per cui non solo è possibile, ma doveroso insegnare divertendo.
Grazie a questi criteri l'opuscolo finisce persino per travalicare gli scopi a prima vista limitati al sistema ZX80. Questo, in ultima analisi, per molti aspetti finisce per essere uno degli strumenti atti a rendere concreto il discorso, ma, mutatis mutandis, le idee espresse possono abbastanza agevolmente essere riciclate su qualunque altro elaboratore personal o orientato didatticamente.

Indice generale

CAPITOLO 1

Struttura del calcolatore

Le parti componenti un calcolatore elettronico, vedi fig. 1 sono, in generale, le seguenti:

- unità centrale (CPU)
- unità di ingresso (INPUT)
- unità di uscita (OUTPUT)
- memoria secondaria

Una elaborazione con il calcolatore consiste sempre in una trasformazione di dati. I dati di ingresso vengono elaborati dal calcolatore e trasformati nei dati di uscita.

Le parti componenti il calcolatore ZX80 sono:

- l'unità centrale (CPU)
- l'unità di ingresso, che è una tastiera sensibile al tocco.

L'unità di uscita è un qualunque schermo TV (la televisione di casa), e la memoria secondaria è una cassetta magnetica su un registratore (quello di casa).

L'unità centrale del calcolatore ZX80 è formata da:

- il microprocessore D780C - 1
- la memoria a sola lettura, ROM (Read Only Memory)
- la memoria per lettura e scrittura, RAM. (Random Access Memory)

Il microprocessore D780C - 1 comprende:

- l'unità di controllo, che controlla lo svolgimento delle istruzioni del programma;
- l'unità aritmetico/logica, che esegue le operazioni aritmetiche e i controlli logici;
- alcuni registri speciali, usati come memoria di lavoro dal microprocessore

Ogni calcolatore viene costruito con la capacità di svolgere un gruppo di istruzioni, che costituisce il *linguaggio macchina del calcolatore*. Una opportuna se-

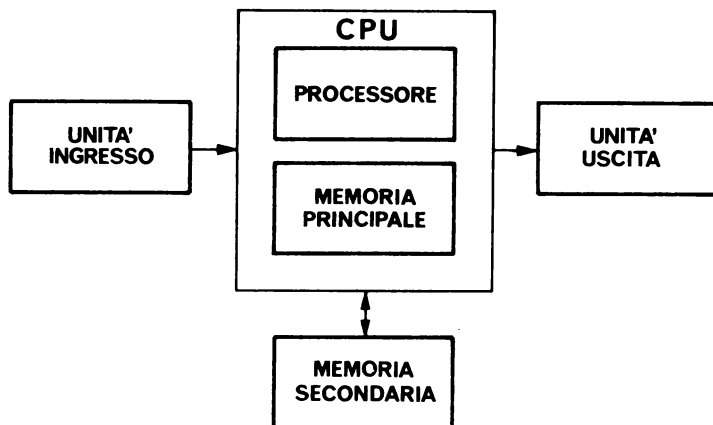



Fig. 1 - Struttura del calcolatore

quenza di istruzioni in linguaggio macchina, costituisce un *programma* per il calcolatore. Il programma si memorizza nella memoria del calcolatore e l'unità centrale, opportunamente avviata, è capace di prelevare automaticamente le istruzioni del programma dalla memoria e di eseguirle una dopo l'altra.

La programmazione in linguaggio macchina risulta abbastanza difficile. Per questa ragione sono stati messi a punto dei linguaggi di programmazione di facile apprendimento per l'uomo, e si è pensato di fare svolgere al calcolatore il lavoro di tradurre tali linguaggi nel linguaggio macchina. Uno dei linguaggi più diffusi e di più facile apprendimento è il BASIC (da Beginners All-purpose Symbolic Instruction Code).

Il calcolatore ZX80 consente all'utente l'uso di una implementazione ridotta del Basic. Perché sia possibile all'utente "parlare in Basic" con lo ZX80 è necessario che nella memoria del calcolatore siano memorizzati i programmi per:

- accettare le frasi del linguaggio BASIC,
- interpretarle e tradurle in linguaggio macchina,
- rendere il più possibile semplice l'uso del calcolatore.

Tali programmi esistono e costituiscono il *Sistema Operativo* e *l'Interpretatore Basic* dello ZX80; essi sono forniti dalla casa costruttrice già memorizzati in forma stabile nella memoria a sola lettura *ROM.*

La memoria per lettura e scrittura *RAM* serve per memorizzare i programmi scritti dall'utente, i dati ed i risultati. La memoria RAM è labile, cioè si cancella, quando l'utente lo desidera e comunque quando si spegne il calcolatore. Per questa ragione si adopera la memoria secondaria, costituita dalla cassetta magnetica, per registrare programmi e dati in modo permanente.

CAPITOLO 2

Il programma

Un programma è una sequenza ordinata di istruzioni il cui significato deve essere chiaro sia a chi le prepara che a chi le riceve. Nel caso dei calcolatori, chi riceve le istruzioni è una macchina programmata e quindi capace di fare solo una serie ben definita di operazioni, niente di più. Per questa ragione la sequenza delle istruzioni per il calcolatore deve essere preparata con cura. Il calcolatore non possiede la fantasia ed il buon senso con cui un essere umano può interpretare delle istruzioni incomplete ricevute da un altro.

Prima di pensare alla stesura di un programma per il calcolatore, si deve esaminare il problema che si vuole risolvere, esponendolo in modo chiaro, evidenziando i dati di partenza e specificando le caratteristiche dei risultati che si desidera ottenere. La procedura con la quale si arriva ai risultati, partendo dai dati iniziali, prende abitualmente il nome di *algoritmo*. Anche quando si risolvono dei problemi manualmente si usano algoritmi risolutivi. Se si riflette su quanto si fa, quando si risolve manualmente un problema, si vede che, in generale, vale lo schema seguente:

1. si scrivono in una zona del foglio i dati iniziali;
2. si eseguono in sequenza delle operazioni aritmetiche;
3. a seconda dei risultati ottenuti si operano delle scelte sul tipo di operazioni con cui proseguire;
4. si ripetono un certo numero di volte dei gruppi di operazioni;
5. si scrivono in una zona del foglio i risultati ottenuti.

Tutti i linguaggi di programmazione mettono a disposizione del programmatore istruzioni adatte per svolgere le operazioni elencate nello schema precedente.

Le situazioni logiche nelle quali si opera possono essere schematizzate nelle tre seguenti:

a) sequenza,
b) diramazione,
c) iterazione.

Si usa descrivere i metodi risolutivi usati nei programmi per il calcolatore mediante dei disegni convenzionali, che prendono il nome di *"diagrammi a blocchi"*.

Si consideri il semplice problema: "Leggere un numero dall'esterno e stabilire se è maggiore di 57". Il diagramma, o schema, a blocchi che descrive il metodo risolutivo del problema posto è quello che segue nella fig. 2.

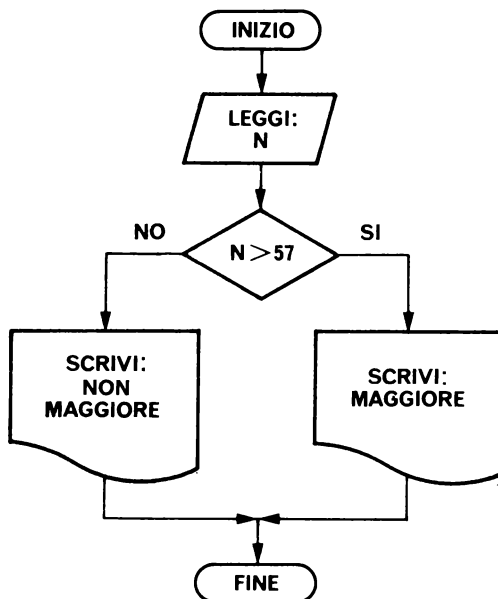
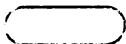
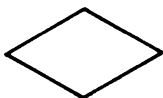



Fig. 2 - Diagramma a blocchi

Come si vede, si usano dei simboli grafici, collegati da segmenti orientati, che danno il senso di percorrenza dello schema. I simboli grafici usati qui sono:

All'interno dei simboli grafici si scrivono sinteticamente le operazioni da svolgere.

Si consideri ora il problema: "Leggere 3 numeri A, B, C e calcolare la media M dei 3 numeri". Il diagramma a blocchi è riportato ora in figura 3.

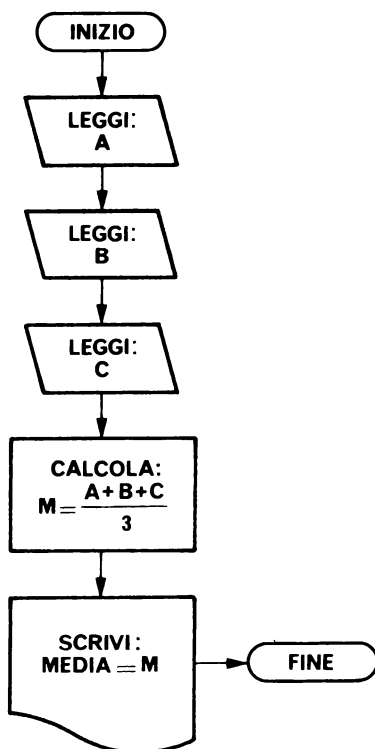



Fig. 3 - Diagramma a blocchi

In questo caso si è usato anche il simbolo grafico rettangolare; esso indica una qualunque operazione di calcolo

Si consideri ora il problema: "Leggere 10 numeri e calcolare la media M dei 10 numeri letti. Il diagramma a blocchi è riportato nella fig. 4.

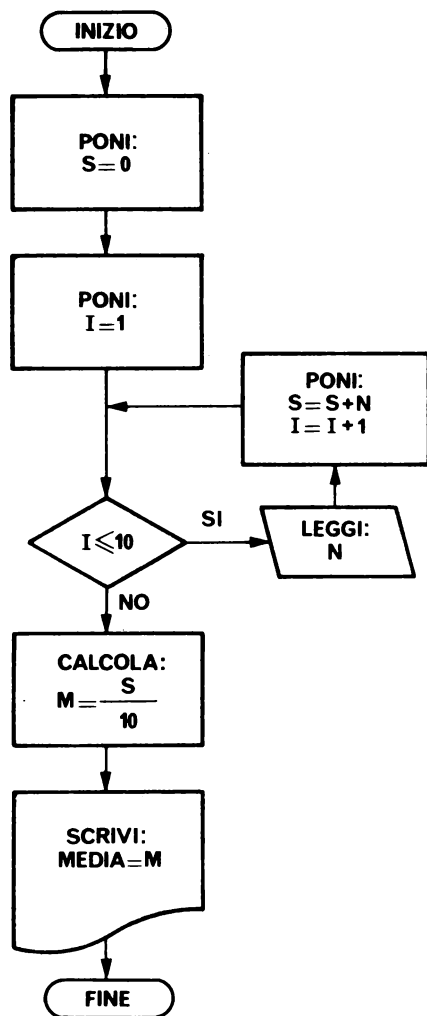
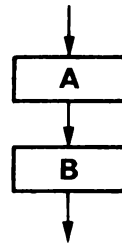
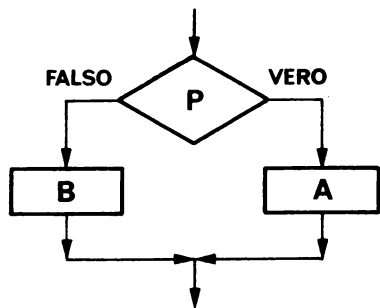
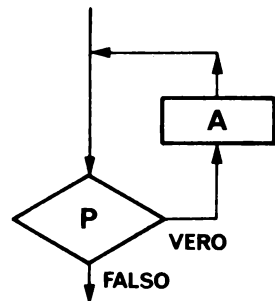



Fig. 4 - Diagramma a blocchi

Si è chiamata S la variabile usata per realizzare la somma dei 10 numeri letti. L'operazione di lettura è iterativa ed il numero delle iterazioni è controllato dal contatore I che viene:

— inizialmente posto a 1,
— controllato per vedere se è minore o uguale a 10,
— incrementato di 1 ad ogni iterazione.

Si riconsiderino ora i 3 diagrammi a blocchi delle figure 2, 3 e 4 con riferimento alle situazioni logiche nelle quali si è operato. Nel diagramma di figura 3 si ha sempre la situazione logica di *"sequenza"*. Nel diagramma di figura 2 si hanno le situazioni logiche di *"sequenza"* e *"diramazione"*. Nel diagramma di fig. 4 si hanno le situazioni logiche di *"sequenza"* e *"iterazione"*.

Queste tre situazioni si possono schematizzare così:

Si possono considerare queste 3 strutture, come strutture di base nella programmazione; loro caratteristica comune è di avere un solo punto di entrata ed un solo punto di uscita.

Il solo diagramma a blocchi non è sufficiente a descrivere il metodo risolutivo di un problema. Esso, in generale, deve essere accompagnato da tutte quelle annotazioni che servano a rendere chiara la situazione in cui si opera.

Quando il problema è stato sufficientemente approfondito ed è disponibile:

- l'analisi del problema,
- il diagramma a blocchi,
- l'elenco delle variabili,

si può passare alla codifica del programma in un linguaggio adatto per il calcolatore sul quale il programma deve essere provato. La codifica viene portata avanti seguendo il diagramma a blocchi ed avendo a disposizione tutte le note relative alle variabili da usare. Terminata la codifica, se non lo si è già fatto, si devono preparare dei casi prova tali da garantire l'esattezza del programma. Si arriva così alla fase di prova del programma; si trovano gli errori ed a seconda dei tipi di errori si interviene nella fase opportuna del lavoro per porvi rimedio. Terminato il programma, cioè quando tutto funziona, si devono preparare le norme operative per l'uso del programma.

Ogni programma per il calcolatore deve essere documentato, cioè deve esistere:

- il testo del problema,
- l'analisi del problema,
- il diagramma a blocchi,
- l'elenco di tutte le variabili usate,
- l'elenco dei casi prova per provare il programma,
- la lista del programma,
- le norme operative per l'uso del programma.

È chiaro che in questo manualetto si fanno degli esempi molto semplici, per i quali la documentazione è necessariamente limitata.

CAPITOLO 3

Installazione del calcolatore ZX80

Lo ZX80 è composto da due unità:

1. il calcolatore,
2. l'alimentatore.

L'alimentatore deve dare 9 Volts in corrente continua a 600 mA non stabilizzati. Il cavo di collegamento termina con uno spinotto Jack del diametro di 3,5 mm, col positivo collegato alla punta. Si osservi il diagramma della figura 5 che riporta i collegamenti.

Guardando il retro del calcolatore si vedono da sinistra a destra 3 prese nere per spinotti Jack, la cui nomenclatura è riportata al di sotto del calcolatore. Il loro utilizzo è:

- TO RECORDER MIC, ingresso microfonico del registratore,
- TO RECORDER EAR, uscita cuffia del registratore,
- 9 V DC IN, spinotto Jack dell'alimentatore.

Proseguendo verso destra, si vede in centro una presa per spinotto Plug americano, destinato al collegamento col Video.

Ancora più a destra si vede una larga fessura destinata all'inserimento della memoria aggiuntiva.

Come video può essere usato un qualunque apparecchio televisivo, sia in bianco e nero che a colori. Si selezioni la banda UHF (quella del secondo canale) e si sintonizzi sul canale 36. Si abbassi il volume al minimo, dato che non esistono uscite sonore. L'uscita sul video è predisposta per dare un quadro di 24 linee di 32 caratteri ciascuna. Si colleghi, utilizzando il cavo in dotazione, l'uscita video dello ZX80 con l'ingresso dell'antenna del televisore. Nel caso il televisore abbia due ingressi a doppio spinotto per l'antenna, sarà necessario munirsi di un adattatore di impedenza 75/300 Ohm e di un cavo adeguato, con relativi spinotti e collegarlo all'ingresso UHF.

A questo punto si accenda il televisore e, quando questo si è scaldato, dopo aver inserito lo spinotto dell'alimentatore (con attenzione!) nella presa giusta

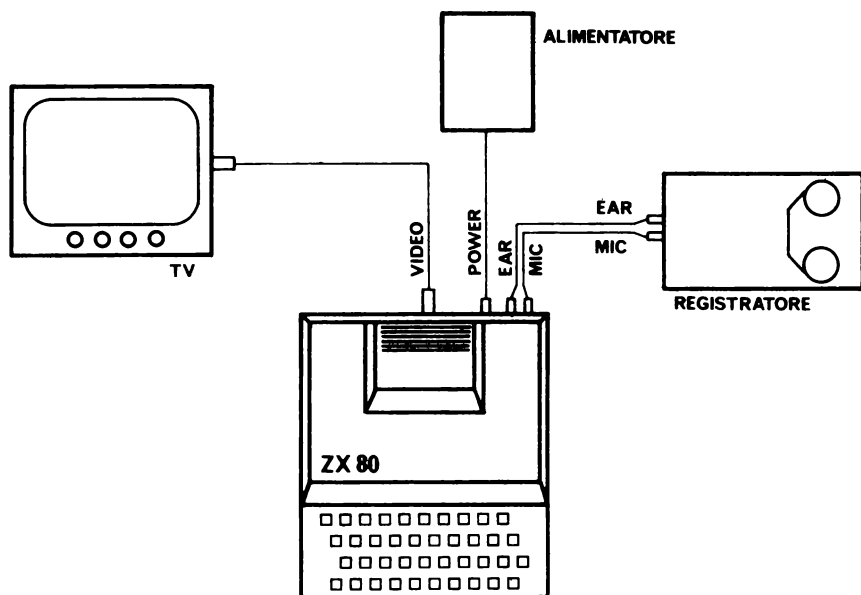



Fig. 5 - Schema di collegamento

(9 V DC IN), si accenda lo ZX80 collegando l'alimentatore alla rete. Quindi si aggiusti la sintonia fino a vedere lo schermo tutto bianco (o grigio chiaro) con nell'angolo a sinistra in basso un quadratino nero contenente la lettera K in bianco. L'immagine deve essere assolutamente stabile. In caso l'immagine non sia buona, si provi a regolare la luminosità ed il contrasto del televisore, ed a sintonizzare il quadro. La lettera K all'interno del quadratino deve essere chiaramente visibile.

Ora il calcolatore ZX80 è in grado di funzionare. Si può eseguire il test riportato nell'inserto per controllare il corretto funzionamento del calcolatore.

Si osservi che nell'angolo in basso a sinistra compare K in campo nero (si dice K in campo inverso), esso indica la posizione del cursore dello schermo e lo stato del calcolatore. Si premano i tasti nella sequenza indicata e si controllino i risultati sullo schermo.

Si può procedere ora al collegamento del registratore, per completare il sistema. Può essere impiegato qualunque tipo di registratore sia a bobina che a cassette, sia stereofonico che monofonico. L'unica condizione necessaria è che il registratore sia dotato di un ingresso per microfono separato e di una uscita per auricolare o cuffia. In dotazione si ha un doppio cavetto con 4 spinotti Jack di diametro 3,5 mm. Questo cavetto può essere usato per il collegamento con lo ZX80.

sinclair ZX80

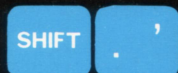
PROGRAMMA TEST

Dopo aver collegato correttamente il vostro ZX80 al TV (si veda a tale proposito il Capitolo 3 a pag. 17) eseguite, con la sequenza indicata, il seguente "Programma Test" per verificarne il corretto funzionamento.

TASTO	SIGNIFICATO
1	Il cursore rimane K in campo inverso ed entra 1.
FOR F	Because the cursor was in K mode, this single Dato che il cursore era in stato K, entra la parola chiave FOR seguita da uno spazio ed il cursore passa allo stato L.
I (	Dato che il cursore è nello stato L entra la lettera I.
SHIFT L =	Holding the SHIFT key while pressing L = Tenendo premuto lo SHIFT mentre si preme il tasto entra il carattere =.
1	Entra 1.
TO SHIFT 4	Tenendo premuto lo SHIFT mentre si preme il tasto entra TO seguito da uno spazio.
9	Entra 9.
NEW LINE	La linea 1 viene accettata come linea di programma e passa nella parte alta dello schermo. il cursore torna nello stato K.

2 — Entra 2.

PRINT O) — Entra PRINT seguito da uno spazio ed il cursore passa allo stato L.

I (— Entra il carattere I.

SHIFT . ' — Tenendo premuto SHIFT entra il carattere,.

NEW LINE — La linea 2 viene accettata.

3 — Il cursore era ritornato nello stato K, entra 3.

NEXT N < — Entra NEXT, seguito da uno spazio.

I (— Il cursore era nello stato L ed entra il carattere I.

NEW LINE — La linea 3 viene accettata.

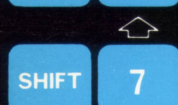
RUN R — Il cursore era ritornato allo stato K entra RUN.

NEW LINE — Per effetto di questo tasto viene accettato il comando RUN e viene eseguito il programma formato dalle 3 istruzioni scritte. Sullo schermo appaiono i numeri da 1 a 9 in quattro colonne.

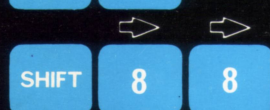
In basso a destra compare 0/3 a indicare che il programma ha terminato la sua esecuzione alla linea 3 con 0 errori.

Premere un tasto qualunque, di seguito appare la lista del programma con il puntatore di linea alla linea 3.

Muove il puntatore di linea su.

Muove il puntatore di linea giù.

Fa ritornare il puntatore sulla linea 2.

Appare una copia della linea 2 in basso sullo schermo, con il cursore dello schermo dopo il numero di linea e la linea può essere modificata.

Sposta il cursore verso destra di un carattere o di una parola chiave.

Sposta il cursore dopo la ,.

Cancella il carattere a sinistra del cursore.

Sposta il cursore a sinistra di I.

Inserisce 2.

Inserisce l'asterisco tra 2 e I.

Fa accettare la nuova versione della linea 2 al posto della vecchia.

Entra RUN.

Fa eseguire il nuovo programma e sullo schermo appaiono in colonna i numeri pari da 2 a 18 con ancora 0/3 in basso a sinistra.

Si colleghi l'uscita MIC dello ZX80 con l'ingresso per microfono (marcato MIC o REC) sul registratore e l'entrata EAR dello ZX80 con l'uscita per auricolare (marcata EAR o MONITOR) del registratore. È importante familiarizzarsi con questi collegamenti perché durante l'uso del registratore andranno fatti e disfatti più volte con sicurezza.

Se il registratore non ha l'ingresso per il microfono e l'uscita per l'auricolare adatti agli spinotti Jack 3,5 mm, sarà necessario munirsi di un adattatore.

Dopo essersi accertati che il registratore è in buone condizioni di funzionamento (testine pulite e, se possibile, smagnetizzate) si può procedere come segue:

1. registrare un programma che si trovi in memoria sul nastro,
2. leggere in memoria un programma che si trovi sul nastro.

Prova 1

— scrivere NEW NEW LINE;
— scrivere 10 REM STO PROVANDO A REGISTRARE NEW LINE;
— mettere il registratore in grado di registrare la voce con i collegamenti al calcolatore staccati;
— avviare il nastro per registrare;
— registrare parlando PROVA DI REGISTRAZIONE e fermare il nastro;
— inserire il collegamento MIC (o REC) tra calcolatore e registratore;
— riavviare il nastro;
— premere subito sulla tastiera SAVE e NEW LINE.

A questo punto si vede scomparire la scrittura dallo schermo, esso diventa grigio, poi si vedono comparire delle righe orizzontali ed alla fine ricompaiono le scritte di prima, attendere 10 secondi e fermare il registratore.

Il programma è stato registrato sul nastro.

Prima di fare la seconda prova si deve azzerare lo schermo e la memoria premendo NEW e poi NEW LINE; si vedrà ricomparire il K nel quadratino nero in fondo al video a sinistra. Riavvolgere il nastro al numero di giri prima della registrazione.

Prova 2

— staccare i collegamenti registratore/calcolatore;
— cercare sul nastro la frase: PROVA DI REGISTRAZIONE;
— dopo la frase si sente un BRR... e poi silenzio; fermare il registratore appena inizia il silenzio;

— inserire il collegamento EAR (o MONITOR) tra registratore e calcolatore;
— riavviare il nastro e premere subito LOAD e NEW LINE;
— lo schermo diventa grigio e poi appare la lista del programma;
— fermare il registratore.

Se le due prove non hanno dato buon esito ritentare seguendo con precisione le istruzioni.

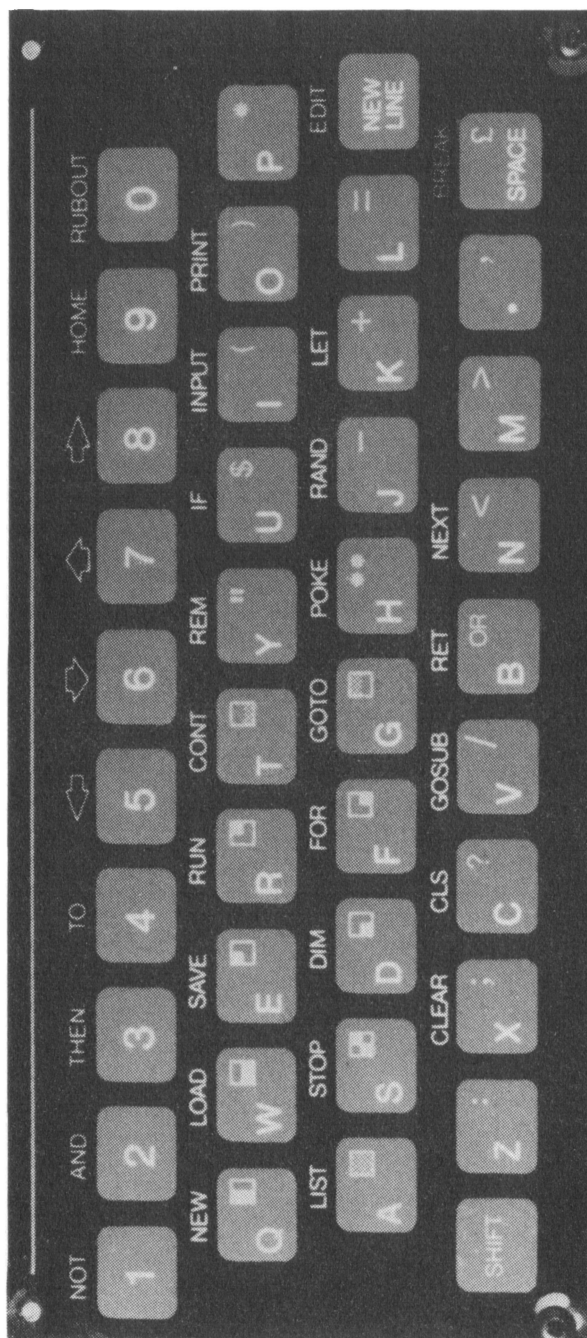
CAPITOLO 4

La tastiera dello ZX80

Osservando la tastiera del calcolatore si vede che alcuni tasti hanno una sola funzione, scritta in bianco all'interno, mentre sopra il tasto è riportata una parola o un simbolo grafico. A questo gruppo appartengono i tasti: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, NEW LINE.

Per attivare la funzione scritta sopra il tasto, in questo caso si deve tenere premuto il tasto SHIFT. Il tasto SHIFT ha una sola funzione: attivare lo SHIFT. Quasi tutti gli altri tasti hanno due funzioni scritte all'interno, una in bianco e una in giallo ed inoltre una funzione scritta sopra il tasto.

Se il calcolatore è nello stato K, rilevabile dal cursore in campo inverso dello schermo, premendo un tasto senza SHIFT si attiva la funzione scritta sopra; mentre se il calcolatore è nello stato L, rilevabile dal cursore dello schermo, premendo un tasto senza SHIFT si attiva la funzione scritta in bianco all'interno del tasto. La funzione scritta in giallo all'interno del tasto, si attiva, per questo gruppo, premendo lo SHIFT contemporaneamente al tasto.

Per usare la tastiera il movimento delle dita deve essere delicato ed i tasti non devono essere battuti come sulle macchine da scrivere. È importante imparare a distinguere la lettera O dallo zero. Sulla tastiera lo zero si trova in alto a destra dopo il 9 ed è meno rotondo della lettera O che si trova nella fila sotto.

Tastiera dello ZX80

CAPITOLO 5

Il linguaggio BASIC per il calcolatore ZX80

Il programma è formato da linee numerate da 1 a 9999 al massimo; ogni linea contiene una istruzione. La numerazione progressiva delle linee rappresenta anche l'ordine di esecuzione delle istruzioni del programma. Si usa numerare le linee del programma con numeri non consecutivi, in tale modo è possibile fare delle inserzioni di linee senza dover rinumerare tutte le altre. Si può usare: 10, 20, 30, ecc.

Le *istruzioni* sono formate dalle *parole chiave* proprie del linguaggio e dai *simboli* inventati dal programmatore per indicare gli *operandi*. Gli operandi possono essere *costanti* o *variabili*.

Le *costanti* possono essere:

- numeri interi, con segno, compresi tra − 32768 e + 32767;
- stringhe di caratteri alfanumerici delimitate dagli apici che ovviamente non possono far parte della stringa. Es.: "OGGI PIOVE". Sono lunghe a piacere.

Le *variabili* possono essere:

- variabili numeriche intere
- variabili stringhe alfanumeriche

Le *variabili numeriche intere* hanno nomi simbolici che devono sempre iniziare con una lettera e possono contenere, dopo il primo carattere, sia cifre numeriche che lettere, ed essere lunghi a piacere. Naturalmente, più il nome è lungo più spazio occupa in memoria, quindi è meglio limitarsi nel numero dei caratteri dei nomi delle variabili. Le variabili numeriche possono contenere numeri compresi tra − 32768 e + 32767. Sono nomi validi per questo tipo di variabili: PAGA, I, A1, A2, A3, SCONTO, ecc.

Le *variabili stringa* hanno nomi simbolici formati da una lettera seguita dal carattere $ (dollaro) e possono contenere stringhe di qualsiasi numero di caratte-

ri (compatibilmente con la capacità della memoria). Dato che le lettere dell'alfabeto sono 26, in un programma si possono avere al massimo 26 stringhe. Nomi validi sono: A$, G$, ecc.

Si possono avere *variabili numeriche con indice*, cioè gruppi di variabili, rappresentate globalmente dallo stesso nome, e distinte tra loro da un indice. In questo caso il nome delle variabili può essere formato da una sola lettera. Es.: A(I), dove A è il nome del gruppo di variabili ed I è la variabile che funge da indice del gruppo.

Le *variabili di controllo* (vedi istruzioni FOR/NEXT) devono avere il nome formato da una sola lettera.

Le variabili in programmazione sono da intendersi come "contenitori di dati" e quindi per le variabili hanno senso operazioni del tipo di quelle viste nel Capitolo 2, come I = I + 1. Tali operazioni in un testo di matematica sono considerate errate.
Le *variabili singole* in BASIC vengono definite quando compaiono la prima volta a sinistra di un =, cioè quando gli viene assegnato un valore. Per esempio scrivere I = I + 1, senza avere scritto prima I = qualcosa, dà errore, perché I compare a destra dell'uguale e quindi dovrebbe essere già stata definita. Le variabili con indice, vengono definite quando si usa la frase Basic di dimensionamento.

Gli operandi delle istruzioni possono anche essere delle *espressioni* formate da costanti, variabili e operatori. Gli operatori aritmetici sono:

1. elevamento a potenza (**)
2. negazione unitaria (-)
3. moltiplicazione (*)
4. divisione (/)
5. somma (+) e sottrazione (−)

inoltre nelle espressioni si possono usare le parentesi. L'ordine di valutazione di una espressione è da sinistra a destra, eseguendo prima le operazioni racchiuse tra parentesi e rispettando la priorità sopra elencata per gli operatori.
Oltre agli operatori aritmetici, già visti, esistono anche gli operatori relazionali e gli operatori logici, e possono essere usati nelle espressioni.

Gli *operatori relazionali* sono:

1. = uguale (SHIFT e L)
2. > maggiore (SHIFT e M)
3. < minore (SHIFT e N)

e vengono usati per analizzare il verificarsi di queste tre condizioni tra due diverse espressioni.

Gli *operatori logici* sono:

1. NOT negazione (SHIFT e 1)
2. AND una e l'altra (SHIFT e 2)
3. OR una oppure l'altra (SHIFT e B)

e servono per costruire relazioni condizionali più complicate di quelle che si possono formare usando solo gli operatori relazionali. L'ordine di precedenza per valutare le condizioni di relazione quando compaiono gli operatori logici è il seguente: NOT, AND, OR.

Esistono inoltre le variabili logiche che sono il risultato di una *espressione relazionale* del tipo:

1. maggiore
2. minore
3. uguale
4. o altro usando gli operatori logici

e possono avere il valore VERO o il valore FALSO: VERO corrisponde al valore − 1 e FALSO al valore 0.

Le *istruzioni BASIC* possono essere divise in 5 gruppi:

1. comandi di sistema,
2. istruzioni di controllo,
3. istruzioni di ingresso e uscita dei dati,
4. istruzione di assegnazione,
5. istruzioni varie e di servizio.

Una buona parte delle istruzioni possono essere usate sia come istruzioni di programma che come comandi da eseguire in modo immediato.

Nella descrizione delle istruzioni si farà riferimento a:

- modo immediato di esecuzione, senza memorizzazione dell'istruzione,
- modo differito di esecuzione, dopo la memorizzazione dell'istruzione nel programma.

1. Comandi di sistema

Le istruzioni di questo gruppo sono comandi di sistema e sono eseguiti prevalentemente in modo immediato, possono essere inseriti in un programma per esecuzione differita, ma questo non risulta molto utile. Essi sono:

NEW	azzera la memoria dello ZX80 e predispone la ricezione di un nuovo programma.

RUN — azzera tutte le variabili del programma presente in memoria e ne fa partire l'esecuzione della linea con il numero di linea minore. Si può anche scrivere RUN n, dove n è un numero di linea, ed allora dopo aver azzerato le variabili, l'esecuzione parte dalla linea n.

LIST — lista sul video il programma presente in memoria; se il programma supera le 22 linee, vengono listate solo le prime 22 linee, infatti lo schermo tiene 24 linee, ma dopo la lista del programma esce una riga in bianco ed una riga è occupata dal cursore dello schermo. Se si vuole ottenere la lista della parte restante del programma si può scrivere LIST n, allora si ottiene:
- se la linea n è già sullo schermo, compare il puntatore di linea a marcarla;
- se la linea non è già sullo schermo ed n riferisce la linea successiva o quella dopo, lo schermo scorre ed aggiunge la o le due linee, con il puntatore di linea a marcare la linea richiesta;
- se n indica una linea oltre l'ultima presente più due, compare il pezzo di programma dalla linea precedente la n in avanti, sempre con il puntatore di linea alla linea richiesta.

LOAD — trasferisce un programma registrato sul nastro magnetico in memoria, esattamente nello stato in cui era stato registrato. La memoria viene azzerata anche se non si è dato il comando NEW.

SAVE — memorizza sul nastro magnetico il programma che si trova in memoria.

2. Istruzioni di controllo

Fanno parte di questo gruppo le istruzioni che permettono di uscire dalla situazione logica di svolgimento sequenziale, per numero di linea crescente, delle linee del programma. Sono:

STOP — è una istruzione da usare solo in modo differito. Causa la fermata del programma al numero di linea dello STOP. Può essere utile per vedere dei risultati parziali del programma, chiedendone la stampa in modo immediato. Per proseguire nell'esecuzione si può usare il tasto CONT che corrisponde alla parola chiave CONTINUE e fa proseguire l'esecuzione dalla linea successiva allo STOP.

CONTINUE — è una istruzione che ha senso usare solo in modo immediato

per fare proseguire il programma dopo una fermata per STOP.

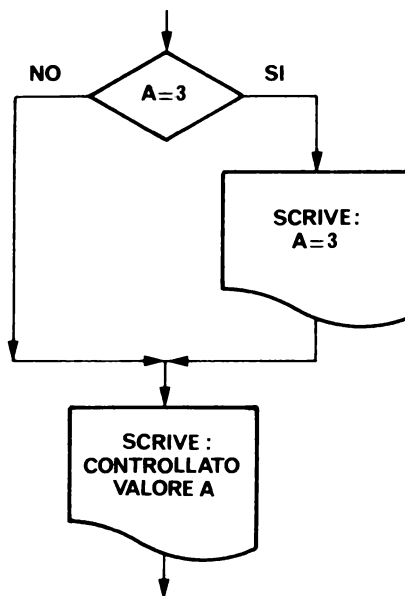
GOTO n — è l'istruzione di salto senza condizioni. Può essere usata in modo immediato per far proseguire l'esecuzione di un programma dalla linea n, anche in fase di partenza iniziale, solo che non vengono azzerate le variabili del programma. Si usa in modo differito nel programma per fare proseguire l'esecuzione dalla linea n. Questo n può essere:
— un numero di linea,
— una variabile numerica,
— una espressione aritmetica intera.

IF condizione THEN istruzione — è una istruzione da usare solo in modo differito. Serve per la condizione logica di diramazione.

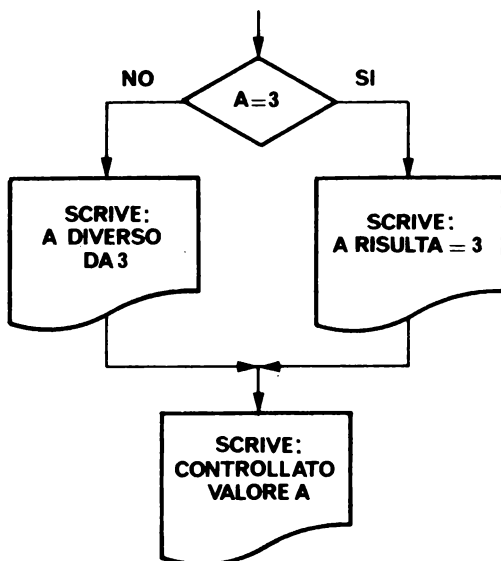
Esempi di frasi IF...THEN...

100 IF A = 3 THEN PRINT "A = 3"
120 PRINT "CONTROLLATO VALORE A"

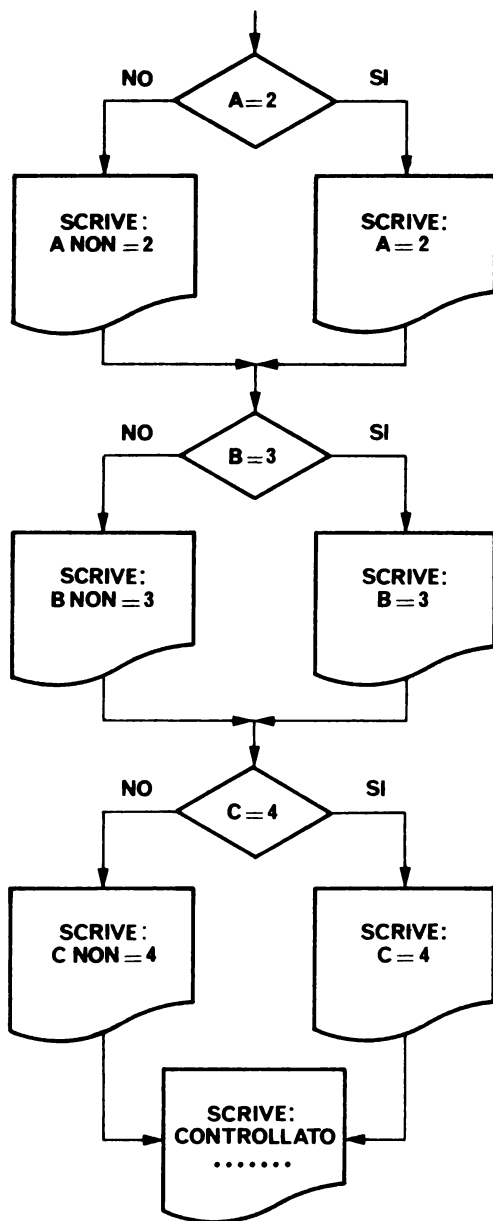
Se A = 3 il programma quando arriva alla linea 100 esegue l'istruzione dopo il THEN e quindi scrive A = 3, dopo prosegue con la linea 120 e quindi scrive CONTROLLATO VALORE A; se A non è uguale a 3, va direttamente alla linea 120 e scrive solo CONTROLLATO VALORE A. Quindi viene eseguita l'istruzione dopo THEN se la condizione risulta vera; viene eseguita solo la linea seguente se la condizione risulta falsa.

Le istruzioni 100 e 120 corrispondono al diagramma a blocchi pubblicato a lato.

100 IF A = 3 THEN GOTO 150
120 PRINT "A DIVERSO DA 3"
130 PRINT "CONTROLLATO VALORE A"
140 GOTO .....
150 PRINT "A RISULTA = 3"
160 GOTO 130

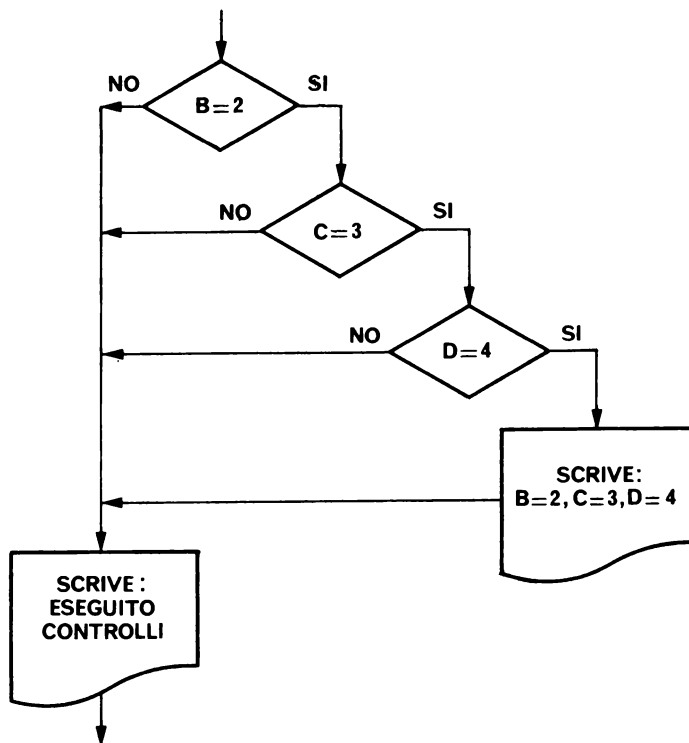
Se A = 3 il programma prosegue dalla linea 150, scrive A RISULTA = 3 e poi ritorna alla linea 130 e scrive CONTROLLATO VALORE A e poi ....; se A non uguale a 3, il programma segue alla linea 120 e scrive A DIVERSO DA 3 e poi prosegue e scrive CONTROLLATO VALORE A e poi...

Le istruzioni 100 ÷ 160 corrispondono al seguente diagramma a blocchi.

```
100 IF A = 2 THEN GOTO 130
110 PRINT "A NON = 2"
120 GOTO 140
130 PRINT "A = 2"
140 IF B = 3 THEN GOTO 170
150 PRINT "B NON = 3"
160 GOTO 175
170 PRINT "B = 3"
175 IF C = 4 THEN GOTO 200
180 PRINT "C NON = 4"
190 GOTO 210
200 PRINT "C = 4"
210 PRINT "CONTROLLATI VA-
    LORI A, B, C"
```

se A = 2 va a 130, scrive A = 2
se B = 3 va a 170, scrive B = 3
se C = 4 va a 200, scrive C = 4
poi scrive CONTROLLATO...
se A NON = 2, scrive A NON = 2
e va a controllare B
se B NON = 3, scrive B NON = 3
e va a controllare C
se C NON = 4, scrive C NON = 4
e va a scrivere CONTROLLATO...

Le istruzioni 100÷210 corrispondono al seguente diagramma a blocchi.

```
100 IF B = 2 THEN IF C = 3 THEN
    IF D = 4 THEN PRINT "B = 2, C
    = 3, D = 4"
120 PRINT "ESEGUITO CONTRO-
    LLI"
```

Se B = 2, C = 3 e D = 4 il programma scrive: B = 2, C = 3, D = 4 e dopo scrive: ESEGUITO CONTROLLI mentre se una delle condizioni non è verificata scrive solo: ESEGUITO CONTROLLI.

Vale il diagramma a blocchi che segue:

Questa situazione viene chiamata degli *IF nidificati*, si vedrà che può essere programmata anche in altro modo. Essa risulta più complicata delle situazioni logiche di diramazione viste fino ad ora.

Sfruttando le relazioni con operatori logici la 100 può essere scritta così:

```
100 IF B = 2 AND C = 3 AND D = 4 THEN PRINT "B = 2, C = 3, D = 4"
```

e dal punto di vista logico nel diagramma si avrebbe un solo rombo con scritte all'interno tutte le condizioni che devono essere analizzate.

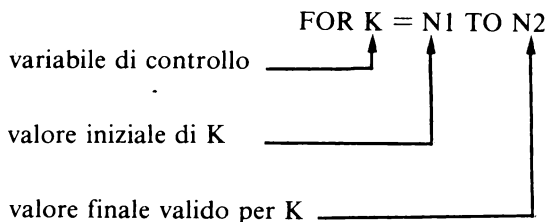
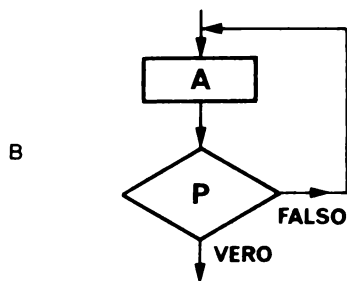
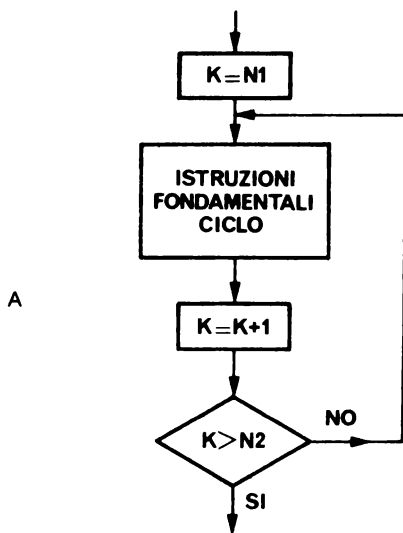
IF...THEN

si sono visti diversi esempi di uso della istruzione IF ... THEN; si può riassumerne il comportamento così: se la condizione analizzata risulta vera viene eseguita l'istruzione dopo THEN, se la condizione risulta falsa il programma prosegue dalla linea seguente.

FOR K = N1, TO N2
.....
.....
NEXT K

sono due istruzioni da usare sempre in coppia ed in modo differito. Servono per realizzare la condizione logica di iterazione. Le parole chiave sono FOR...TO e poi NEXT. K è la variabile che controlla le iterazioni, e si è già detto, che le variabili di controllo devono avere un nome formato da una sola lettera. N1 ed N2 possono essere dei numeri interi, delle variabili intere e delle espressioni intere; comunque deve risultare N1 minore di N2. N1 ed N2 possono essere anche negativi.

Quando il programma incontra la linea con FOR, viene posto K = N1; infatti N1 è il valore iniziale della variabile di controllo. Poi vengono eseguite le linee di programma tra FOR e NEXT. Quando il programma arriva alla linea con NEXT, prima incrementa di 1 la variabile di controllo K e poi controlla se ha superato il valore N2; se il valore N2 non è stato superato il programma torna ad eseguire le linee comprese tra FOR e NEXT, tali linee vengono dette *operazioni fondamentali del ciclo*. Se il valore N2 è stato superato il programma prosegue con la linea seguente la linea del NEXT. Da quanto detto si vede che un ciclo FOR/NEXT viene percorso almeno una volta anche se N1 è uguale o supera N2 in partenza. Inoltre si vede che al momento dell'abbandono del ciclo la variabile di controllo K risulta uguale ad N2 + 1.

FOR K = N1 TO N2

Il diagramma a blocchi della operazione iterativa in questa implementazione del BASIC si presenta pertanto come raffigurato nella pagina seguente, fig. A.

Questo diagramma risulta un po' diverso dalla situazione logica iterazione già presentata, si può considerare corrispondente ad una situazione logica derivata da quella già vista e rispondente allo schema di fig. B.

la cui spiegazione è la seguente: esegui A fino a quando la condizione P è falsa.

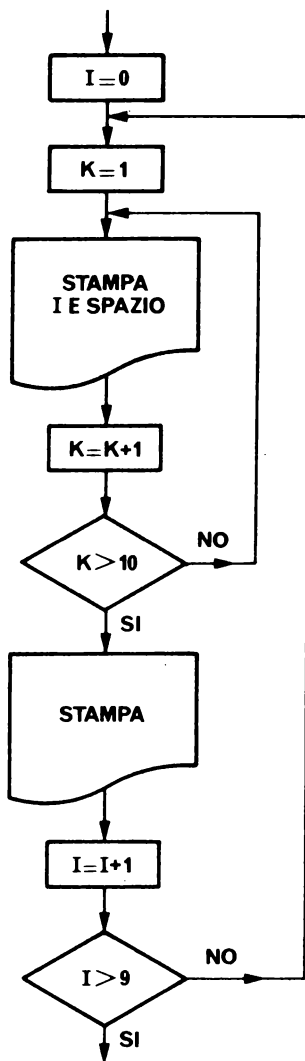
Il concetto di operazione ciclica è fondamentale nella programmazione, si fa però notare che se non fosse disponibile la coppia di istruzioni FOR/NEXT si potrebbe ottenere lo stesso risultato gestendo a programma un contatore di ciclo.
Si può fare uso di cicli nidificati, cioè uno interno all'altro. Si veda il seguente esempio: si voglia evidenziare sullo schermo una tabella di 10 righe e di 10 colonne, nella quale la prima riga contiene tutti zeri, la seconda tutti 1, ecc.
Si possono scrivere le seguenti istruzioni:

```
10 FOR I = 0 TO 9
20 FOR K = 1 TO 10
30 PRINT I; " ";
40 NEXT K
50 PRINT
60 NEXT I
```

Il programma opera come segue:

— la linea 10 apre un ciclo FOR controllato dalla variabile I ponendo I = 0;
— la linea 20 apre un secondo ciclo FOR, interno al precedente, controllato dalla variabile K, ponendo K = 1;
— la linea 30 è l'istruzione fondamentale del ciclo controllato da K e stampa la variabile I seguita da uno spazio;

— la linea 40 incrementa K e controlla se ha superato il valore 10; se $K \leqslant 10$ torna alla linea 30, se $K > 10$ prosegue alla linea 50;
— la linea 50 stampa una riga a vuoto per andare a capo;
— la linea 60 incrementa I e controlla se ha superato il valore 9; se $I \leqslant 9$ torna alla linea 20 per riiniziare un nuovo ciclo per la prossima riga, se $I > 9$ il programma è terminato

Il diagramma a blocchi esplicativo della procedura è quello che segue:

Seguono alcuni schemi di cicli FOR/NEXT nidificati corretti:

Lo schema che segue invece non è corretto:

Del gruppo delle istruzioni di controllo fanno parte anche le istruzioni GOSUB e RETURN che verranno trattati a parte nel capitolo dedicato ai sottoprogrammi.

3. Istruzioni di ingresso e uscita dei dati

INPUT nome-variabile quando il programma incontra questa istruzione si ferma in attesa di dati dalla tastiera. Se nome-variabile è il nome di una variabile numerica si vedono sullo schermo i due caratteri L ed S in campo inverso, quando si scrive la prima cifra del numero scompare S, mentre L scompare con tutto il numero quando si preme NEW LINE per fare accettare il numero scritto. Se nome-variabile è il nome di una variabile stringa si vede il carattere L in campo inverso compreso tra gli apici delimitatori di stringa.

I numeri interi per essere accettati devono essere compresi tra − 32768 e + 32767. Le stringhe possono essere lunghe a piacere pur di non superare la zona di memoria che il sistema mette a disposizione per memorizzarle. Se mentre si scrive una stringa scompare il carattere L in campo inverso e l'apice di chiusura significa che si è superato lo spazio disponibile. In questo caso si possono cancellare dei caratteri con SHIFT e RUBOUT fino a veder ricomparire il puntatore e gli apici. L'istruzione INPUT può essere usata solo in modo differito.

PRINT lista di variabili, numeri stringhe questa istruzione può essere usata sia in modo immediato che differito. I dati da stampare sono separati o da virgola o da punto e virgola.

I due separatori hanno effetto diverso:

- la virgola fa posizionare alle colonne 9, 17 e 25 dello schermo, non si possono fare più di 4 zone di stampa;
- il punto e virgola fa stampare i dati come sono senza caratteri separatori.

Usando la virgola come separatore, la riga del video viene divisa in 4 zone di 8 caratteri ciascuna. Se in una zona il dato è più lungo di 7 caratteri, la virgola ha l'effetto di far saltare tutta la zona di stampa seguente. Due virgole vicine fanno saltare due zone di stampa.

Se la linea da stampare supera i 32 caratteri si ha automaticamente la continuazione della stampa sulla prossima riga. Si può evitare di andare a capo dopo la stampa (se i caratteri sono meno di 32) facendo terminare la lista di variabili da stampare o con una virgola o con un punto e virgola; l'effetto del separatore continua con la stampa seguente.

Se i numeri sono negativi viene stampato il segno meno. Nella lista dei dati da stampare possono comparire anche espressioni; esse vengono calcolate e viene stampato il risultato.

4. Istruzione di assegnazione

La forma di questa istruzione è:

LET variabile = espressione	essa consente di svolgere calcoli e di trasferire dati da una variabile ad un'altra.

Le regole per il calcolo delle espressioni sono già state viste; il risultato dell'espressione viene assegnato alla variabile che si trova a sinistra del simbolo uguale. Può essere usata sia in modo immediato che differito.

5. Istruzioni varie e di servizio

CLEAR	azzera tutte le variabili del programma, può essere usata sia in modo immediato che differito.
CLS	azzera lo schermo, può essere usata sia in modo immediato che differito.
REM	indica che quanto segue sulla linea è un commento, serve per inserire annotazioni in un programma che non influiscono sullo svolgimento del programma stesso; non ha senso usarla in modo immediato.
DIM nome-variabile valore massimo indice	serve per dimensionare una variabile con indice.

Il nome-variabile può essere solo una singola lettera. Viene predisposta una variabile con indice formata da tanti elementi quanto è il valore massimo dell'indice + 1, infatti l'indice varia da zero al valore massimo. Es.: DIM A(10) predispone la variabile con indice A che comprende 11 elementi.

Nel programma gli elementi del gruppo vengono individuati per mezzo dell'indice; l'indice deve essere una variabile numerica o una espressione numerica. Può essere usata sia in modo immediato che differito.

RANDOMISE predispone il punto di partenza della sequenza dei numeri a caso, ottenibili con la funzione RND, ad un numero uguale al valore del contatore dei fotogrammi dello schermo TV. Se si scrive RANDOMISE n si predispone il punto di partenza della sequenza a n. Può essere usata sia in modo immediato che differito.

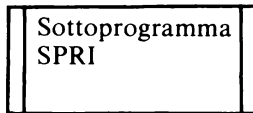
POKE a,b scrive nel byte di indirizzo a l'espressione b. Naturalmente l'espressione b (in particolare il numero b) deve avere valore minore di 256. Può essere usata sia in modo immediato che differito. Il contatore dei fotogrammi TV sta nei byte di indirizzo 16415 (cifre più significative) e 16414. Con la POKE si può azzerare il contenuto dei due byte per calcolare, mediante una lettura successiva, un intervallo di tempo.

CAPITOLO 6

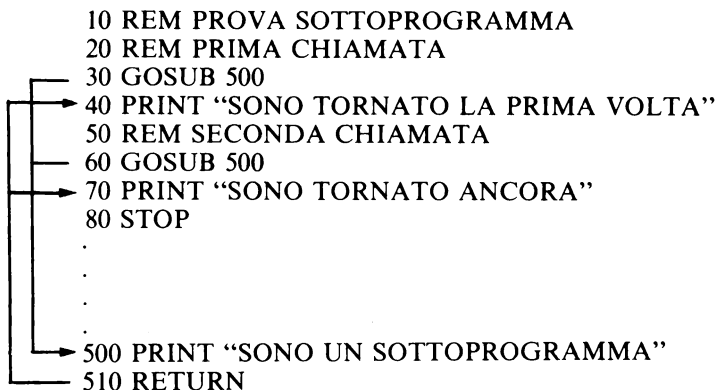
I sottoprogrammi

Qualora in un programma si presenti la necessità di ripetere più volte una stessa sequenza di istruzioni, è conveniente organizzare tale sequenza come sottoprogramma interno al programma. In questo modo si diminuisce il numero di righe totale del programma. È necessario allora che il linguaggio di programmazione fornisca:

- una istruzione per saltare all'inizio del sottoprogramma interno, memorizzando il numero di linea successivo a quello della linea che contiene l'istruzione di salto;
- una istruzione con la quale chiudere il sottoprogramma interno e ritornare alla sequenza principale al numero di linea già memorizzato.

Nella stesura dei diagrammi a blocchi si usa questo simbolo grafico,

per indicare la chiamata ad un sottoprogramma, scrivendo all'interno il nome del sottoprogramma chiamato. A parte si traccia il diagramma a blocchi del sottoprogramma.

Lo schema di programma che segue può esemplificare cosa è un sottoprogramma:

```
10 REM PROVA SOTTOPROGRAMMA
20 REM PRIMA CHIAMATA
30 GOSUB 500
40 PRINT "SONO TORNATO LA PRIMA VOLTA"
50 REM SECONDA CHIAMATA
60 GOSUB 500
70 PRINT "SONO TORNATO ANCORA"
80 STOP
.
.
.
.
500 PRINT "SONO UN SOTTOPROGRAMMA"
510 RETURN
```

L'istruzione:

GOSUB numero-linea serve per: — saltare al sottoprogramma;
— memorizzare il numero di linea per il ritorno

L'istruzione:

RETURN deve chiudere logicamente il sottoprogramma e serve per ritornare al programma chiamante.

Se si prova il programma esempio appena visto si vedrà apparire sullo schermo:

SONO UN SOTTOPROGRAMMA
SONO TORNATO LA PRIMA VOLTA
SONO UN SOTTOPROGRAMMA
SONO TORNATO ANCORA

e la sequenza di esecuzione delle linee di programma è la sequente:

10-20-30-500-510-40-50-60-500-510-70-80

Le istruzioni GOSUB e RETURN hanno senso solo se usate in modo differito.
È possibile per un sottoprogramma chiamare un altro sottoprogramma e, se la cosa ha un significato logico corretto, richiamare anche se stesso.

CAPITOLO 7

Le funzioni implementate dal BASIC

Esistono 8 funzioni già implementate nell'interprete BASIC; esse si possono chiamare *funzioni intrinseche*. I nomi di queste 8 funzioni sono riportati in un rettangolo nero in alto a destra sopra la tastiera del calcolatore. Per usarle il programmatore deve scriverne il nome per intero usando i normali tasti. Si passa ad elencarle spiegandone il significato.

RND(n) — fornisce un numero pseudo-random compreso tra 1 e n. Al posto di n può essere usata anche una espressione numerica. Se il valore dell'espressione è positivo si ottiene un numero pseudo-random compreso tra 1 ed il valore dell'espressione. Se il valore è zero si ottiene 1. Se il valore è negativo si ottiene un numero pseudo-random compreso o tra −32768 e il valore meno 1, o tra 1 e 32767. Ogni volta che viene usata la funzione RND il sistema usa un generatore di numeri a caso che produce una sequenza fissa di numeri, anche se tale sequenza è molto lunga. Se prima di usare la funzione RND si usa l'istruzione RANDOMISE, viene usato come numero di partenza nella sequenza un numero uguale al valore del contatore dei fotogrammi dello schermo TV. Tale contatore inizia a contare quando viene acceso il sistema e viene incrementato di 1 ogni 50esimo di secondo. Il valore di questo contatore può essere alterato usando la istruzione POKE. Usando RANDOMISE si ottiene una diversa sequenza di numeri ogni volta che si usa il programma. Se invece si usa RANDOMISE n, con n diverso da zero, si ottiene di fare partire la sequenza dei numeri pseudo-random da n e quindi, ogni volta che si usa il programma, si ottiene la stessa sequenza di numeri.

ABS (espressione) — fornisce il valore assoluto dell'espressione.

PEEK(n) — fornisce il contenuto della locazione di memoria di indirizzo n. La memoria dello ZX80 è indirizzabile a byte, mentre le variabili intere sono contenute in due bytes consecutivi. Analogamente sono contenute in due bytes consecutivi molte variabili usate dal sistema operativo, come il contatore dei fotogrammi dello schermo. Se

una variabile è contenuta in due bytes consecutivi di memoria, le sue cifre più significative si trovano nel byte di indirizzo dispari e le meno significative nel byte di indirizzo pari numericamente precedente. Se per esempio si vuole il valore decimale del contatore dei fotogrammi TV, si opera così:

10 LET C = PEEK (16414) + 256 PEEK (16415)

20 PRINT C

infatti 16414 e 16415 sono gli indirizzi dei due bytes usati per tale contatore.

USR(n) — permette di andare ad eseguire un pezzo di programma (routine) che si trovi memorizzato a partire dall'indirizzo di memoria n. Tale programma deve essere scritto in codice macchina. Questa funzione fornisce il valore della coppia di registri del sistema HL, se a causa del calcolo iniziato in n il valore di tali registri è stato modificato, oppure l'indirizzo n, se il valore dei registri HL non è stato modificato. Prima di usare questa funzione si deve avere una completa padronanza del sistema.

Le rimanenti 4 funzioni permettono di operare sulle stringhe alfanumeriche; si passa ad elencarle spiegandone il significato.

CHR$(x) — dove x è un numero, una variabile intera o una espressione numerica. Questa funzione fornisce il carattere corrispondente al codice rappresentato da x. Nella Appendice-A è riportata una tabella dei codici del sistema.

TL$ (stringa-alfanumerica) — questa funzione ritorna la stringa senza il suo primo carattere. Esempio:

10 PRINT TL$ ("ABCDE") mostra allo schermo BCDE

10 PRINT TL$(G$) se G$ contiene PIOVE, mostra allo schermo IOVE.

CODE (stringa-alfanumerica) — fornisce il codice numerico corrispondente al primo carattere della stringa. Tra le parentesi si può mettere una stringa tra apici o il nome di una variabile stringa. Esempio:

10 PRINT CODE ("OGGI PIOVE") mostra allo schermo 52 che è il codice numerico della lettera O.

STR$(n) — fornisce una stringa di caratteri equivalente al valore dell'espressione n. Esempio:

10 LET A$ = STR$ (4567) pone A$ = "4567"

10 LET A$ = STR$(X) se X contiene il numero 1227, pone A$ = "1227"

CAPITOLO 8

Norme operative

Sullo schermo si hanno due indicatori. Uno è il *cursore dello schermo* ed è rappresentato da un quadratino nero con una lettera in bianco; si dice per questo in campo inverso. La lettera può essere:

- K per indicare che è in attesa di comando,
- L per indicare che è in attesa di carattere.

Il cursore si sdoppia, cioè compaiono due cursori, in caso di errore o di attesa di dati numerici. In questo caso il nuovo cursore contiene la lettera S. Il cursore dello schermo sdoppiato nei casi di errore, si pone con la parte S prima dell'errore e l'altra L dopo. Nel caso di attesa di dato numerico la due parti stanno vicine con L prima di S. Durante l'immissione di un programma il cursore lavora nella parte bassa dello schermo e la sua posizione può essere modificata dall'utente per mezzo dei due tasti freccia-a-sinistra (SHIFT e 5) e freccia-a-destra (SHIFT e 8). Quando lavora un programma, e vengono eseguite le operazioni di INPUT, il cursore dello schermo sale nella posizione attualmente libera e segnala l'attesa di un numero con le due parti LS e l'attesa di una stringa con "L".

L'altro indicatore è il *puntatore di linea* ed è rappresentato da un quadratino nero con in bianco il simbolo di maggiore $>$. Questo puntatore segnala l'ultima linea di programma scritta durante il caricamento di un programma. La posizione del puntatore di linea può essere modificata usando i tasti freccia giù (SHIFT e 6) e freccia-su (SHIFT e 7).

Anche il tasto HOME (SHIFT e 9) agisce sul puntatore di linea facendolo salire alla linea 0. Dal momento che la linea zero non esiste sullo schermo, usando HOME, il puntatore di linea svanisce. Se si vuole far ricomparire il puntatore di linea basta usare il tasto freccia-giù (SHIFT e 6).

Immissione di un programma dalla tastiera

Prima di scrivere un nuovo programma premere: NEW e poi NEW LINE per azzerare la memoria. Il cursore dello schermo si pone a K. Le linee di programma si scrivono usando i tasti appropriati e si vedono formare nella parte bassa

dello schermo; il cursore segue la scrittura della linea, cambiando di stato e segnalando gli errori, come detto sopra. Quando la linea è completa il tasto NEW LINE la fa accettare, solo se non ci sono errori; se sono presenti errori la linea rimane nella parte bassa dello schermo. In questo caso si muove opportunamente il cursore e si cancellano gli errori usando il tasto RUBOUT (SHIFT e 0). Si deve tener presente che RUBOUT cancella quello che è scritto a sinistra del cursore; se si cancella un carattere normale viene cancellato un solo carattere, se si cancella una parola chiave, viene cancellata tutta.

Se si vuole inserire un carattere, basta usare il tasto appropriato ed il carattere viene inserito a sinistra del cursore spostando tutta la linea verso destra. Lo spostamento è di una posizione per inserimento di caratteri normali, di più posizioni per inserimento di parole chiave.

Quando la linea è giusta e viene accettata, essa passa nella parte alta dello schermo nella posizione che le compete in base al numero di linea, con il puntatore di linea dopo il numero. Se nelle lista del programma esisteva già una linea con lo stesso numero della nuova, la vecchia linea viene cancellata ed al suo posto va la nuova linea.

Una linea di programma già accettata può necessitare di correzioni per errori logici o di simboli fatti dal programmatore e non contrastanti con la sintassi del linguaggio. In tale caso si può procedere così:

— si sposta il puntatore di linea alla linea voluta usando i due tasti SHIFT e 6 (↓) oppure SHIFT e 7 (↑),
— si usano i tasti SHIFT e NEW LINE (EDIT), questo fa comparire la linea nella parte bassa dello schermo,
— spostando il cursore dello schermo per mezzo dei tasti SHIFT e 5 (←) oppure SHIFT e 7 (→), usando SHIFT e 0 (RUBOUT) ed i tasti appropriati, si modifica la linea,
— premendo NEW LINE la linea modificata va a sostituire la vecchia nella lista del programma.

Questa procedura di EDIT può essere utilmente impiegata qualora in un programma si abbiano linee di programma uguali a meno del numero di linea o, comunque, abbastanza simili.

Quando il programma supera le 22 linee sullo schermo ad ogni nuova linea aggiunta si ha la perdita apparente delle prime linee. Queste linee scompaiono solo dallo schermo, ma restano in memoria. Per far comparire la lista dall'inizio basta usare il tasto LIST. Il comando LIST è già stato descritto nel capitolo 5. Si ricorda comunque che con LIST si ha la lista dall'inizio per le righe che entrano nello schermo, mentre con LIST n, si ha la lista dalla linea n in avanti.

Se si desidera cancellare una linea di programma, si deve scrivere il numero della linea e subito dopo NEW LINE. Se si scrive il numero della linea seguito da

uno o più spazi e poi NEW LINE, la linea vecchia viene sostituita dalla nuova, contenente solo il numero di linea, e questa non disturba durante l'esecuzione del programma.

Memorizzazione ed immissione di un programma su/da nastro magnetico

Tale procedura è stata già descritta nel capitolo 3.

Esecuzione di un programma

Per eseguire un programma si usa il tasto RUN. L'effetto di RUN è di azzerare tutte le variabili del programma e di mandarlo in esecuzione. Se si vuole mandare in esecuzione un programma senza azzerare le variabili (per esempio un programma letto da nastro e per il quale si vogliono mantenere precedenti risultati) si scrive sulla tastiera GOTO n, dove n è il numero della prima linea del programma. Mentre un programma lavora lo schermo si oscura e scompaiono le scritte. Quando il programma è in attesa di ingresso di dati si ferma ed appare nella posizione libera dello schermo [L][S] per dati numerici; " [L] " per stringhe alfanumeriche. Dopo i dati si preme NEW LINE per farli accettare ed i dati scompaiono, ma restano sullo schermo i risultati dei precedenti PRINT.

Se in risposta a richiesta di dati numerici si danno altri caratteri, con NEW LINE scompare l'INPUT, ma appare in basso sullo schermo la segnalazione di errore. Si può far ripartire il programma dal punto voluto usando GOTO n.

Quando il programma ha terminato il suo lavoro, restano sullo schermo i risultati dei PRINT ed in basso la segnalazione di fine programma. Se si tocca un qualunque tasto riappare la lista del programma. Se nel programma sono stati inseriti degli STOP, si hanno degli arresti nell'esecuzione. Per proseguire si può usare il tasto CONT. Solo che appena si tocca CONT riappare la lista del programma, premendo ancora CONT compare in basso CONTINUE e premendo NEW LINE l'esecuzione prosegue dalla linea seguente lo STOP, solo si è perso il contenuto precedente dello schermo, ma non i risultati precedenti che sono rimasti in memoria.

Dopo uno STOP si può anche proseguire con GOTO n, però anche in questo caso appena si tocca un qualunque tasto ricompare la lista e con GOTO n si prosegue, ma scompaiono le scritte precendenti dallo schermo.

Esecuzione di istruzioni in modo immediato

Nei capitoli precedenti si è segnalato, per ogni istruzione descritta, se può essere usata in modo immediato. Tenendo presente quanto sopra detto, si ricorda che se si scrive una istruzione BASIC, senza premettere il numero di linea,

quando si preme NEW LINE, l'istruzione viene eseguita. Se è presente un programma in memoria, esso non viene disturbato dall'esecuzione di istruzioni in modo immediato; naturalmente con, per esempio, istruzioni di tipo LET, possono essere modificati valori di variabili del programma, e, questo comporta una variazione nei risultati finali.

Uno degli usi più comuni delle istruzioni in modo immediato è quello di usare frasi PRINT agli STOP programmati per vedere sullo schermo dei risultati intermedi.

Quando un programma è abbastanza complicato, si ricorre all'inserzione di un certo numero di STOP nei punti chiave, proprio per poter seguire l'andamento del programma in fase di prova. Una volta che il programma è collaudato gli STOP possono essere eliminati.

Situazioni di emergenza

Potrebbe capitare di scrivere un programma dal quale non si riesce più a uscire, come il seguente:

```
10 INPUT N      chiede un numero
20 PRINT N      stampa il numero
30 GOTO 10      torna a 10
```

In questo caso alla richiesta di N numerico si può rispondere, per uscire dal ciclo, dando una risposta errata, cioè una lettera. Il sistema segnala l'errore e si ferma. Toccando un qualunque tasto ricompare la lista del programma. Se nel programma precedente si cerca di rispondere con un numero più grande di 32767 (massimo accettato per i numeri), il sistema non lo accetta e quindi non serve per uscire dal ciclo.

Se invece il programma ha un ciclo errato dal quale non esce più, e il ciclo non contiene istruzioni di INPUT, per uscire dal ciclo si può usare il tasto BREAK. Questo tasto interrompe l'esecuzione del programma e provoca uno STOP forzato. Si può continuare l'esecuzione con CONT.

In caso di vera emergenza, cioè quando non si sa più cosa fare, si può spegnere il calcolatore, per poi riaccenderlo e continuare. Si ricordi però che spegnendo il calcolatore tutto il contenuto della memoria RAM si perde.

CAPITOLO 9

Errori segnalati dal sistema

Il sistema segnala gli errori facendo apparire nella parte bassa dello schermo a sinistra un codice nella forma n/m, dove: n = numero dell'errore m = numero di linea del programma.

Tabella degli errori

Tipo Istruz.	**Cod. n**	**Significato**
vari	0	Si è usato il tasto BREAK; m rappresenta il numero della linea dopo quella in esecuzione al momento del BREAK. Se m = − 1 oppure m = − 2, è stato eseguito con successo un comando in modo immediato. Può essere m negativo oppure m un numero di linea non esistente nel programma; è stato eseguito un GOTO m. Alla fine di un programma m rappresenta l'ultimo numero di linea presente nel programma andato a buon fine.
NEXT	1	m = numero linea che ha causato l'errore. Esiste un NEXT con una variabile, già definita dal programma, ma che non è la stessa usata nel FOR precedente il NEXT. m = numero della linea che ha causato l'errore.
qualunque	2	È stata usata una variabile che non è stata definita in precedenza. Una variabile singola viene definita con una LET di assegnazione. Una variabile con indice viene definita mediante la DIM. m = numero della linea che ha causato l'errore.
qualunque	3	L'indice di una variabile con indice è fuori dai limiti consentiti dalla DIM o c'è un errore nel calcolo dell'indice. m = numero della linea che ha causato l'errore.

Tipo Istruz.	Cod. n	Significato
LET INPUT DIM PRINT	4	Non c'è più posto per raggiungere una nuova variabile numerica o per aumentare il numero di caratteri di una stringa oppure manca posto sullo schermo.
PRINT	5	Non c'è più posto sullo schermo. Se in questo caso si preme CONT due volte e poi NEW LINE appare sullo schermo la fine della stampa. m = numero di linea che ha causato l'errore.
qualunque	6	Si è avuto supero di capacità durante il calcolo, cioè il risultato è minore di −32768 o maggiore di +32767. In alcuni casi si ha questo errore anche per risultato uguale a −32768. m = numero di linea che ha causato l'errore.
RETURN	7	Si è incontrato un RETURN senza che sia stato preceduto da un GOSUB. m = −2
INPUT	8	Si è tentato di usare l'istruzione INPUT in modo immediato.
STOP	9	m = numero di linea contenente il comando STOP. Se si preme CONT il programma continua dalla linea seguente la m.

Dopo una segnalazione di errore da parte del sistema, a seconda dei casi, si interverrà opportunamente, eventualmente modificando il programma.

CAPITOLO 10

Utilizzo della memoria RAM

La memoria è formata da elementi a due stati; se uno stato è rappresentato dalla cifra 0 e l'altro dalla cifra 1 si ha uno stretto legame con il sistema binario di numerazione. La memoria dello ZX80 è formata da questi elementi raggruppati 8 a 8. Il gruppo di 8 elementi prende il nome di *byte*, ed ogni elemento prende il nome di *bit* da Binary digIT.

La grandezza della memoria si misura in *bytes*. Lo ZX80 standard ha la memoria RAM di 1K bytes. K ha il valore convenzionale di 1024, quindi la memoria RAM dello ZX80 standard è di 1024 bytes, cioè di 1024 gruppi di 8 bits.

Ogni byte è indirizzabile singolarmente. La memoria RAM comincia all'indirizzo 16384 e, se è di 1 solo K, termina all'indirizzo 17407. Se si aggiunge la memoria addizionale di 3K, gli indirizzi della RAM vanno da 16384 a 20479. Ogni byte può contenere un numero che al massimo è formato da 8 cifre 1 consecutive, tale numero corrisponde a 255 nel sistema decimale.

Se si considera un qualunque numero decimale, per esempio: 4318, si vede che esso si può scrivere:

$$4318 = 4 \times 10^3 + 3 \times 10^2 + 1 \times 10^1 + 8 \times 10^0$$

Analogamente se si considera il numero del sistema binario 11111111, si vede che esso si può scrivere:

$$11111111 = 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0.$$

Per confondere tra loro numeri appartenenti a sistemi di numerazione diversi essi si possono scrivere tra parentesi riportando in basso a destra la base del sistema di numerazione usato. Così: $(5)_{10} = (101)_2$.

Nell'aritmetica binaria si fanno regolarmente i calcoli; le regole base sono: 1 + 1 fa 0 con il riporto di 1; 1 + 0 fa 1.

Dal momento che i numeri binari sono difficilmente leggibili si usa interpretarli come appartenenti al sistema esadecimale, di base 16, raggruppando i bits a 4

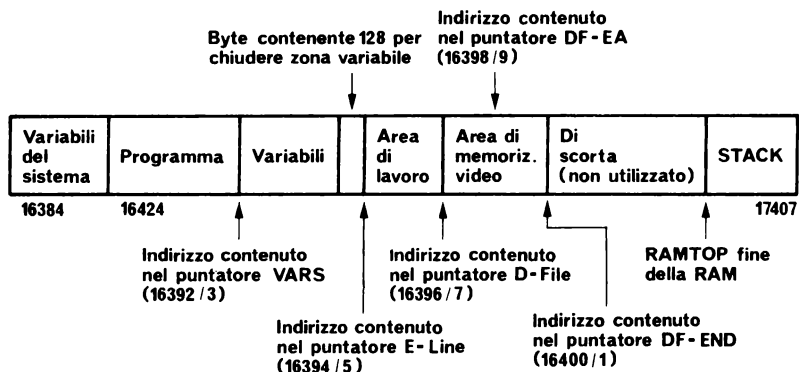
a 4, infatti $2^4 = 16$. In questo modo un byte è formato da 2 cifre esadecimali, di più facile lettura. Nel sistema esadecimale sono necessari 16 simboli diversi per rappresentare i numeri; era ovvio scegliere le cifre da 0 a 9 e poi le prime 6 lettere dell'alfabeto da A ad F. Per questa ragione A corrisponde a 10 decimale, B a 11, C a 12, D a 13, E a 14 ed F a 15. Allora il byte contenente $(255)_{10}$, equivale $(FF)_{16}$ o $(11111111)_2$.

Nello ZX80 i numeri interi sono memorizzati in due bytes consecutivi con le cifre meno significative nel primo byte e le più significative nel secondo byte, ma l'indirizzo del numero (al quale è memorizzato il numero) è quello del primo byte, avente indirizzo pari. Così, per esempio, se all'indirizzo 16600 è memorizzato il numero 3427 si ha: nel byte 16600 la parte meno significativa e cioè 0110 0011, e nel byte 16001 la parte più significativa e cioè 0000 1101. Leggendoli in esadecimale il contenuto di 16000 è 63 e quello di 16001 è 0D.

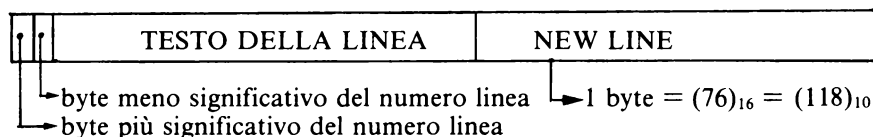
Se si usa la funzione BASIC PEEK per leggere i due bytes e si vuole ricostruire il numero che rappresentano si deve procedere così:

```
10 LET A = PEEK (16000)
20 LET B = PEEK (16001)
30 LET N = B * 256 + A
40 PRINT N
```

I numeri interi hanno sempre il primo bit del byte più significativo a zero se sono positivi. I numeri negativi sono invece memorizzati nella forma del complemento a due e quindi hanno il primo bit del byte più significativo sempre a 1.

La memoria ROM del sistema standard è di 4K bytes ed occupa i bytes da 0 a 4095; quindi prima della memoria RAM sono ancora disponibili indirizzi per 12K di memoria ROM per future espansioni dello ZX80. Analogamente la memoria RAM può essere estesa ancora di 12K raggiungendo l'indirizzo massimo 32767.
Si osservi ora lo schema seguente, nel quale si descrivono i contenuti della RAM.

La prima zona "variabili del sistema" è di dimensioni fisse, va dall'indirizzo 16384 all'indirizzo 16423, occupando 40 bytes. Nell'Appendice B si dà l'elenco delle variabili che la compongono con brevi spiegazioni sul loro significato. A questo gruppo di variabili appartengono quelle citate nello schema precedente, che contengono i puntatori alle diverse zone nelle quali è divisa la RAM. Infatti le diverse aree della RAM non sono di dimensioni fisse, ma variano al variare delle situazioni dei programmi presenti in memoria; per questo è necessario avere delle variabili (fisse) che svolgono la funzione di puntatori all'inizio delle zone.

La zona "programma" inizia sempre all'indirizzo 16424 e termina prima della zona "variabili". L'indirizzo di inizio della zona "variabili" è contenuto nella variabile, chiamata simbolicamente VARS, avente indirizzo 16392 e quindi occupante i due bytes 16392 e 16393. La zona variabili è chiusa da un byte contenente $(128)_{10}$ e quindi $(80)_{16}$ e quindi $(10000000)_2$.

Le linee del programma Basic sono memorizzate nella forma seguente:

Si noti che i numeri di linea sono rappresentati mettendo a sinistra il byte più significativo ed a destra quello meno significativo, esattamente al contrario del comportamento abituale dello ZX80. Dato che i numeri di linea consentiti vanno da 1 a 9999, si vede subito che il byte più significativo di tali numeri ha i due primi bits di sinistra a zero. Come si vedrà tra poco le variabili sono rappresentate in modo tale da non avere mai i primi due bit a zero e quindi l'incontrare un byte, dopo NEW LINE, che segnala sempre la fine di una linea di programma, con uno dei primi due bits diversi da zero, segnala subito che il programma è terminato. Comunque per l'inizio della zona variabili esiste il puntatore VARS.

Le variabili hanno tutte nomi simbolici che iniziano con una lettera, i codici rappresentativi delle lettere vanno da 38 a 63 in decimale e quindi da 26 a 3F in esadecimale e quindi tutte le lettere hanno un codice di 6 bit ed inoltre il primo bit (dei 6) è sempre 1. Come si vede dagli schemi sopra, il sistema gioca sui primi bit delle lettere aggiungendone altri, i primi due, ed eventualmente azzerando il terzo, per distinguere tra loro i diversi tipi di variabili che tratta.

La variabile di nome simbolico E-Line, avente indirizzo 16394 (bytes 16394 e 16395), contiene l'indirizzo di inizio dell'area di lavoro; tale area è usata dal sistema per diverse esigenze.

La variabile di nome simbolico D-File, avente indirizzo 16396 (bytes 16396 e 16397), contiene l'indirizzo di inizio dell'area destinata al video (DISPLAY FILE). Tale area contiene sempre 25 caratteri NEW LINE cioè $(76)_{16}$. Il primo e

Primo byte
Bytes successivi
011
byte meno significativo
byte piu' significativo
Variabile numerica con nome simbolico di un solo carattere
Lettera
Valore
010
00
10
byte meno significativo
byte piu' significativo
Variabile numerica con nome lungo
Lettera
Secondo carattere
Ultimo carattere
Valore
100
caratteri
caratteri
00 00 0001
Stringa alfanumerica
Lettera
Caratteri della stringa
Apici (codice 1)
101
n
byte meno significativo
byte piu' significativo
byte meno significativo
byte piu' significativo
Variabile con indice
Lettera
Numero elemento - 1
Elemento 0
Elemento n
111
byte meno significativo
byte piu' significativo
byte meno significativo
byte piu' significativo
byte meno significativo
byte piu' significativo
Variabile di controllo di un ciclo FOR
Lettera
Valore variabile
Valore limite (dopo TO)
Numero della linea di programma dopo il FOR

l'ultimo bytes sono sempre a $(76)_{16}$ e tra questi vi sono 24 linee da 0 a 32 caratteri ciascuna.

La variabile DF-EA, indirizzo 16398 (bytes 16398 e 16399), contiene l'indirizzo di inizio della parte bassa dello schermo (dove si formano le istruzioni in fase di caricamento di un programma). La variabile DF-END, indirizzo 16400 (bytes 16400 e 16401) contiene l'indirizzo della parte disponibile dello schermo, cioè fino dove si è arrivati a scrivere.

Il registro SP del sistema punta all'area STACK che inizia in fondo alla memoria RAM e viene usata dal sistema come area di memoria temporanea. L'area STACK è usata per indirizzi decrescenti. Se il programma è troppo lungo l'area del video viene diminuita e non si possono vedere sullo schermo tutte le linee. Analogamente, se il programma è troppo lungo si arriva a occupare l'area STACK ed allora si ha l'errore di tipo 4, già visto.

CAPITOLO 11

Esempi di programmi

1. Programma per eseguire la divisione con tre decimali.

(a) Viene richiesto il dividendo e il divisore.

(b) La variabile X contiene il dividendo e la variabile Y il divisore.

(c) Viene calcolata la parte intera Z del quoziente.

(d) Viene calcolato il resto R1. Il primo decimale, D1 è ottenuto moltiplicando il resto R1 per 10 e poi dividendolo per Y.

(e) Viene calcolato il nuovo resto R2. Il secondo decimale D2 è ottenuto moltiplicando il resto R2 per 10 e dividendo poi per Y.

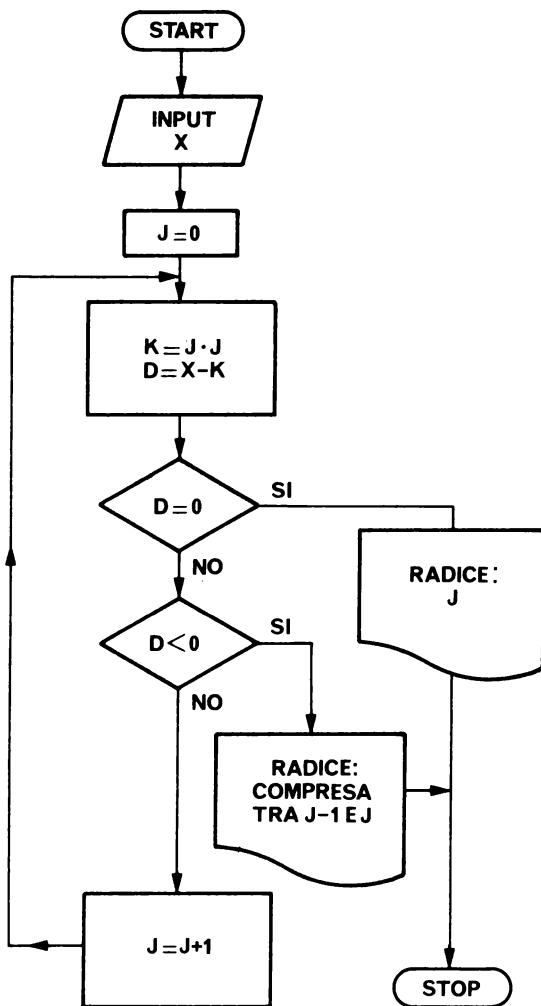
(f) Viene calcolato il nuovo resto R3. Il terzo decimale D3 viene ottenuto moltiplicando il resto R3 per 10 e poi dividendo per Y.

(g) Si stampa il risultato Z.D1D2D3.

```
10 REM DIVISIONE CON 3 DECIMALI
15 PRINT "DIVISIONE CON 3 DECIMALI"
20 PRINT "DIVIDENDO = ?"
30 INPUT X
40 PRINT "DIVISORE =?"
50 INPUT Y
60 LET Z = X/Y
70 LET R1 = X - Z * Y
80 LET D1 = 10 * R1/Y
90 LET R2 = 10 * R1 - D1 * Y
100 LET D2 = 10 * R2/Y
110 LET R3 = 10 * R2 - D2 * Y
120 LET D3 = 10 * R3/Y
130 PRINT "RISULTATO:"; Z; "."; D1; D2; D3
```

2. Programma per calcolare la radice quadrata di un numero

(a) Viene richiesto il numero e memorizzato in X.

(b) Viene inizializzata al valore zero la variabile J, tale variabile viene poi incrementata di 1 ad ogni giro per trovare la radice di X.

(c) Inizia il calcolo ciclico: si calcola K = J * J.
(d) Si calcola D = X − K.
(e) Se D = 0 si stampa J, radice di X e si va allo STOP.
(f) Se D risulta minore di zero, allora non esiste una radice intera esatta e si stampa che la radice è compresa tra J − 1 e J e si va allo STOP.
(g) Se D non risulta minore di zero, si incrementa J di 1 e si torna al punto (c).
(h) Si ferma il programma.

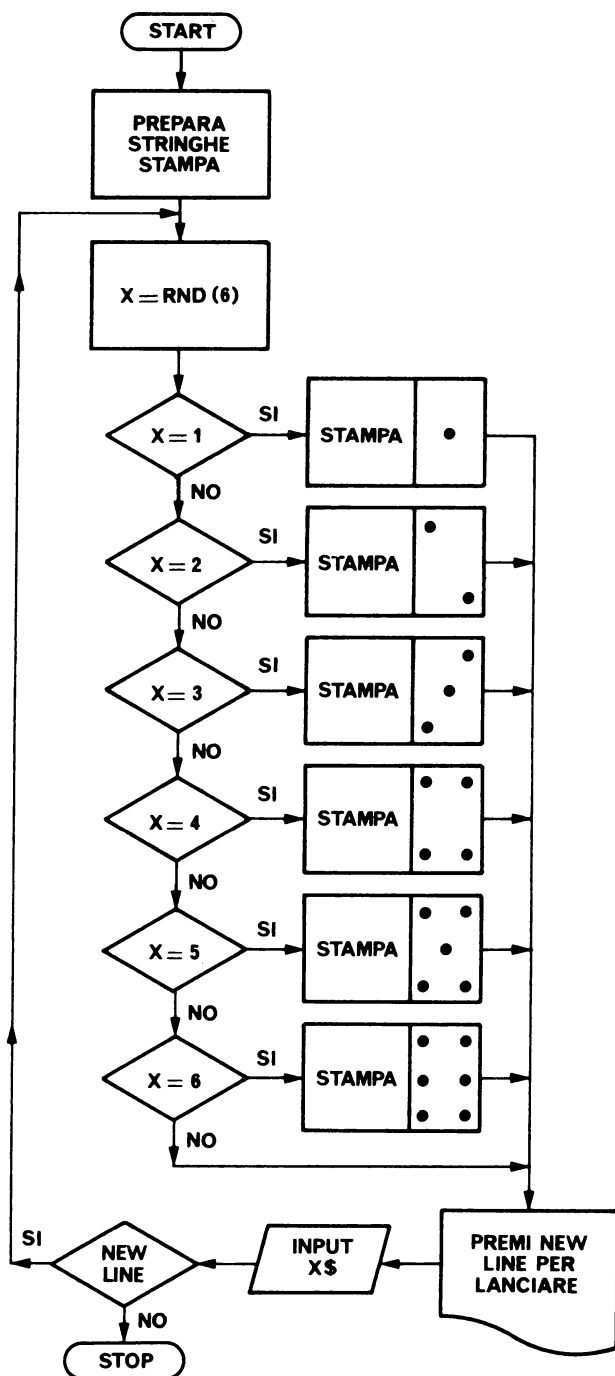
```
10 REM CALCOLO RADICE QUADRATA
15 PRINT "CALCOLO RADICE QUADRATA"
```

```
20 PRINT "SCRIVI IL NUMERO"
30 INPUT X
40 LET J = 0
50 LET K = J * J
60 LET D = X - K
70 IF D = 0 THEN GO TO 110
80 IF D < 0 THEN GO TO 130
90 LET J = J + 1
100 GO TO 50
110 PRINT "RADICE"; J
120 GO TO 140
130 PRINT "RADICE COMPRESA TRA"; (J-1); "E"; J
140 STOP
```

3. Programma per lanciare i dadi

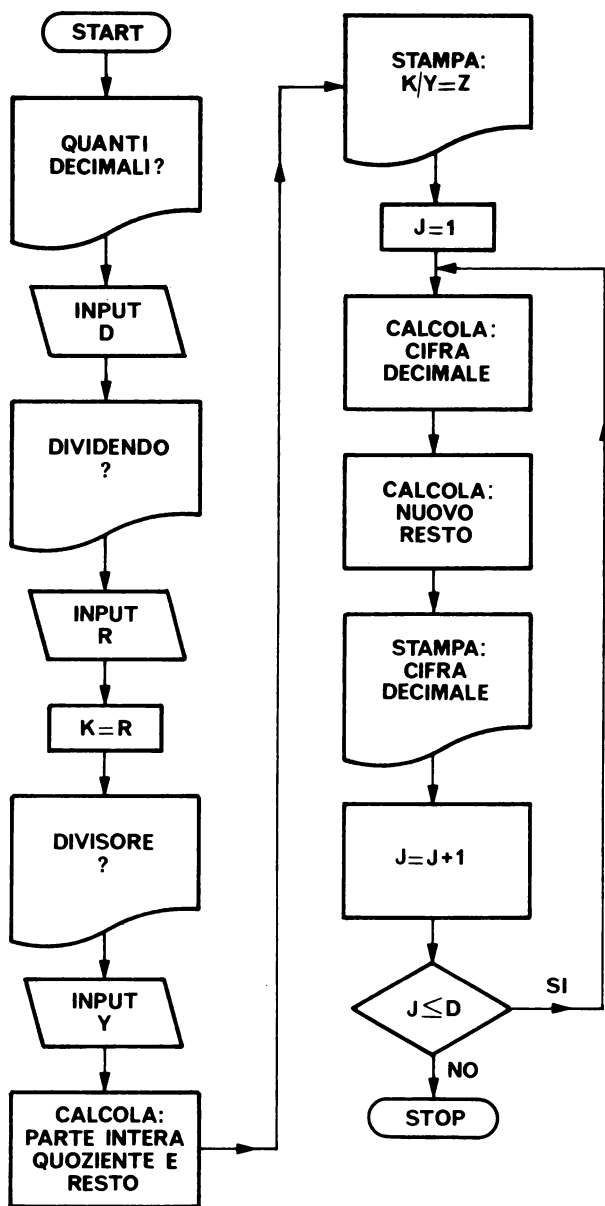
(a) Si preparano delle stringhe contenenti i caratteri grafici necessari per poter evidenziare i dadi sullo schermo.
(b) Si inizia la sequenza di ricerca di un numero pseudo random $\leqslant 6$.
(c) A seconda del numero si salta al pezzo di programma che disegna il dado uscito e poi si torna sempre al punto (d).
(d) Si chiede di premere NEW LINE per lanciare ancora, si analizza i tasto premuto e se è NEW LINE si torna al punto (b) dopo aver azzerato lo schermo, se non si vuole più lanciare si preme un qualunque tasto ed il programma si ferma.

```
10 REM LANCIO DEI DADI
15 PRINT "LANCIO DEI DADI"
20 LET A$ = "
30 LET B$ = "
40 LET C$ = "
50 LET D$ = "
60 LET E$ = "
120 LET X = RND (6)
135 PRINT
136 PRINT
140 IF X = 1 THEN GO TO 200
150 IF X = 2 THEN GO TO 300
160 IF X = 3 THEN GO TO 400
170 IF X = 4 THEN GO TO 500
180 IF X = 5 THEN GO TO 600
190 IF X = 6 THEN GO TO 700
195 GO TO 1000
200 PRINT E$
205 PRINT E$
210 PRINT B$
215 PRINT E$
```

START
PREPARA STRINGHE STAMPA
X = RND (6)
X = 1
SI
NO
STAMPA
X = 2
SI
NO
STAMPA
X = 3
SI
NO
STAMPA
X = 4
SI
NO
STAMPA
X = 5
SI
NO
STAMPA
X = 6
SI
NO
STAMPA
PREMI NEW LINE PER LANCIARE
INPUT X$
NEW LINE
SI
NO
STOP

```
220 PRINT E$
230 GO TO 1000
300 PRINT C$
305 PRINT E$
310 PRINT E$
315 PRINT E$
320 PRINT D$
330 GO TO 1000
400 PRINT D$
405 PRINT E$
410 PRINT B$
415 PRINT E$
420 PRINT C$
430 GO TO 1000
500 PRINT A$
505 PRINT E$
510 PRINT E$
515 PRINT E$
520 PRINT A$
530 GO TO 1000
600 PRINT A$
605 PRINT E$
610 PRINT B$
615 PRINT E$
620 PRINT A$
630 GO TO 1000
700 PRINT A$
705 PRINT E$
710 PRINT A$
715 PRINT E$
720 PRINT A$
1000 PRINT
1001 PRINT
1002 PRINT
1003 PRINT
1010 PRINT "PREMI (NEW LINE) PER LANCIARE"
1011 PRINT "ANCORA"
1100 INPUTS X$
1200 CLS
1300 IF X$ = " " THEN GO TO 120
```

4. Programma per eseguire la divisione con un numero di decimali scelto a piacere

(a)Si richiede il numero di decimali desiderato e si memorizza in D.
(b)Si richiede il dividendo e si memorizza in R.

START
QUANTI
DECIMALI?
INPUT
D
DIVIDENDO
?
INPUT
R
K=R
DIVISORE
?
INPUT
Y
CALCOLA:
PARTE INTERA
QUOZIENTE E
RESTO
STAMPA:
K/Y=Z
J=1
CALCOLA:
CIFRA
DECIMALE
CALCOLA:
NUOVO
RESTO
STAMPA:
CIFRA
DECIMALE
J=J+1
J≤D
SI
NO
STOP

(c) Si pone K = R, cioè K contiene il dividendo.
(d) Si richiede il divisore e si memorizza in Y.
(e) Si calcola Z = R/Y, Z è la parte intera del quoziente; si calcola R = R − Z * Y, cioè si sostituisce al dividendo iniziale il primo resto trovato.
(f) Si scrive la prima parte del risultato senza andare a capo.
(g) Si inizia il ciclo di calcolo per i D decimali.
(h) Si calcola Z = 10 * R/Y cioè si moltiplica il resto per 10 e poi si divide per il divisore; si calcola il nuovo resto e si sostituisce in R al vecchio.
(i) Si stampa Z, cifra decimale calcolata.
(l) Se il ciclo non e finito si torna al punto (h) dopo aver incrementato la variabile J che controlla il ciclo.

```
10 REM DIVISIONE AD ALTA PRECISIONE
15 PRINT "DIVISIONE AD ALTA PRECISIONE"
20 PRINT "QUANTI DECIMALI?"
30 INPUT D
40 PRINT "DIVIDENDO?"
50 INPUT R
55 LET K = R
60 PRINT "DIVISORE?"
70 INPUT Y
80 LET Z = R/Y
90 LET R = R - Z * Y
95 PRINT
100 PRINT K; "/"; Y; "="; Z; ",";
110 FOR J = 1 TO D
120 LET Z = 10 * R/Y
130 LET R = 10 * R - Z * Y
140 PRINT Z;
150 NEXT J
```

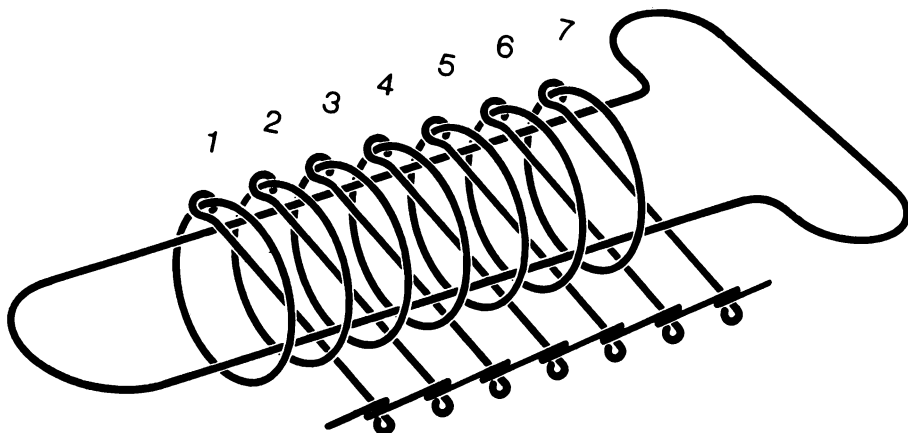
5. Programma per il gioco degli "Anelli Cinesi"

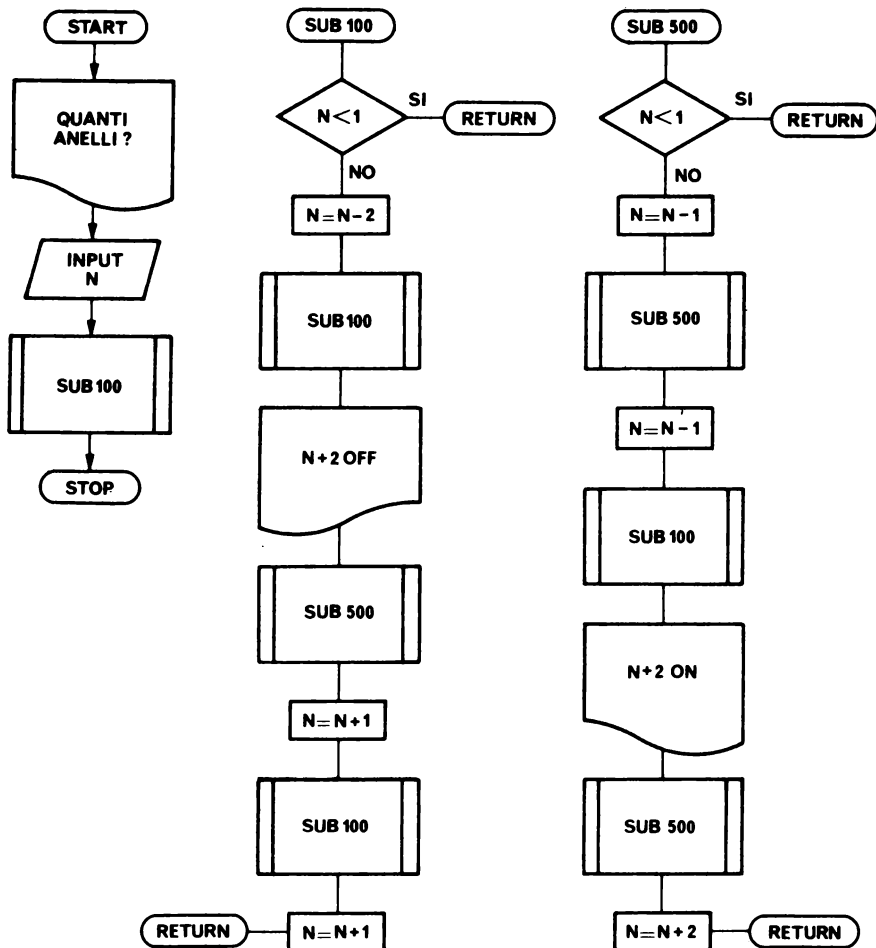
Il gioco consiste nel riuscire a togliere il numero stabilito di anelli, rispettando le seguenti regole:

(a) Si può muovere un anello per volta.
(b) Il primo anello può essere tolto in qualunque momento.
(c) L'anello di posto i (i > 1) può essere tolto o messo se e solo se:

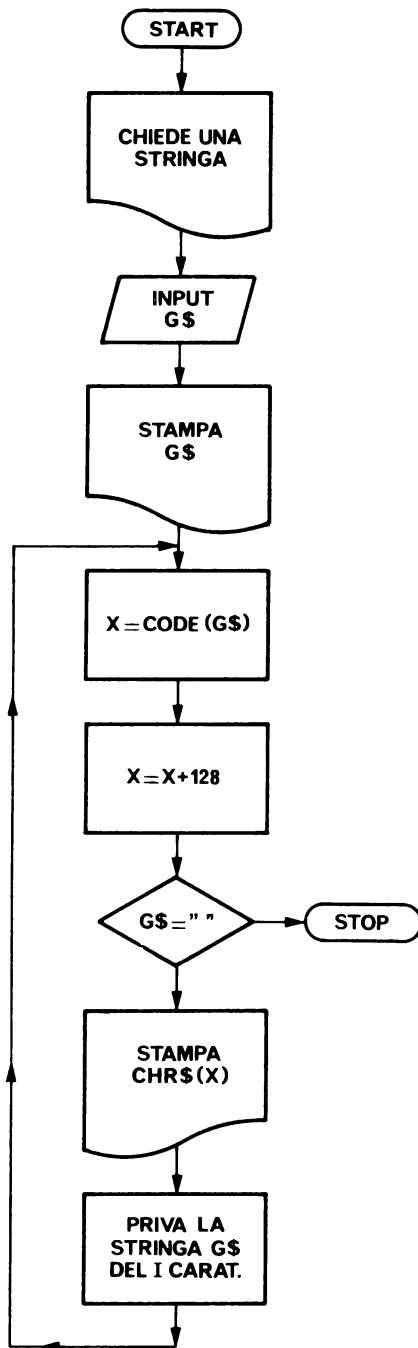
— tutti gli anelli fino al posto i-2 sono stati tolti,
— l'anello di posto i-1 è al suo posto,
— gli anelli di posto > i possono essere in qualunque stato.

Si dice che un anello è *ON* quando è montato, che è *OFF* quando è smontato.

Il programma si articola in un gioco di chiamate a due sottoprogrammi interni

che alternativamente si richiamano o richiamano se stessi. Si ha come output l'elenco delle mosse da fare. Per togliere il settimo anello, le mosse sono molte e non sono contenute tutte nello schermo, è necessario ricorrere al tasto CONT per vederle tutte.

```
10 REM ANELLI CINESI
15 PRINT "ANELLI CINESI"
20 PRINT "QUANTI ANELLI VUOI TOGLIERE?"
25 INPUT N
30 GO SUB 100
40 STOP
100 IF N < 1 THEN RETURN
120 LET N = N - 2
130 GO SUB 100
140 PRINT N + 2; "OFF",
150 GO SUB 500
160 LET N = N + 1
170 GO SUB 100
180 LET N = N + 1
190 RETURN
500 IF N < 1 THEN RETURN
520 LET N = N - 1
530 GO SUB 500
540 LET N = N - 1
550 GO SUB 100
560 PRINT N + 2; "ON",
570 GO SUB 500
575 LET N = N + 2
580 RETURN
```

6. Programma per provare la stampa dei caratteri in campo inverso

(a) Viene chiesta una stringa alfanumerica e memorizzata in G$.
(b) Viene stampata la stringa letta.
(c) Inizia il ciclo di trasformazione dei caratteri componenti la stringa, tale ciclo termina quando si incontra la fine della stringa (stringa nulla, CHR$(1)); ogni carattere viene decodificato, il codice viene modificato aggiungendo 128, e così diventa il carattere in campo inverso, poi viene riconvertito e stampato.

START
CHIEDE UNA
STRINGA
INPUT
G$
STAMPA
G$
X = CODE (G$)
X = X + 128
G$ = " "
STOP
STAMPA
CHR$ (X)
PRIVA LA
STRINGA G$
DEL I CARAT.

```
10 REM PROVA CARATTERI
11 REM IN CAMPO INVERSO
15 PRINT "SCRIVI UN CARATTERE"
20 PRINT "O ALCUNI CARATTERI"
25 INPUT G$
30 PRINT "HAI SCRITTO:" G$
35 PRINT "RISCRIVO IN CAMPO INVERSO"
40 LET X = CODE (G$)
50 LET X = X + 128
60 IF G$ = CHR$(1) THEN GO TO 100
70 PRINT CHR$(X);
80 LET G$ = TL$(G$)
90 GO TO 40
100 STOP
```

7. Programma per tracciare il grafico di due funzioni

Le due funzioni sono Y = X e Z = 24 − X; i valori di Z vengono stampati in nero e quelli di Y in grigio. A seconda del valore della variabile J, che controlla il ciclo più interno, e dei valori di Y e Z viene scelto il carattere da stampare scegliendo tra i 3 seguenti:

```
5 REM GRAFICI DI 2 FUNZIONI
10 LET X = 0
20 PRINT "GRAFICI DI 2 FUNZIONI"
30 PRINT "X ="
40 FOR I = 1 TO 21
50 LET Y = X
60 LET Z = 24 - X
70 PRINT X,
80 FOR J = 1 TO 20
85 IF J > Y AND J = Z THEN PRINT CHR$(3);
90 IF J > Y AND J > Z THEN GO TO 135
95 IF J = Y AND J > Z THEN PRINT CHR$(11);
100 IF J < Y AND J < Z THEN PRINT CHR$(139);
110 IF J < Y AND J > Z THEN PRINT CHR$(11);
115 IF J = Y AND J < Z THEN PRINT CHR$(139);
120 IF J > Y AND J < Z THEN PRINT CHR$(3);
125 IF J = Y AND J = Z THEN PRINT CHR$(139);
130 NEXT J
135 PRINT
140 LET X = X + 1
150 NEXT I
```

8. Programma per misurare la potenza dei riflessi

(a) All'inizio viene creato un ciclo di attesa per rendere possibile l'operazione.
(b) Vengono azzerati i 2 bytes che costituiscono il contatore dei fotogrammi del video.
(c) Viene chiesto di schiacciare NEW LINE.
(d) Viene memorizzata la risposta in C$.
(e) Viene memorizzato il valore dei due byte del contatore.
(f) Viene calcolato il valore del contatore; viene tolto 4 perché si presuppone un ritardo di 80 millisecondi nell'arrivo della risposta. Viene stampato il tempo di risposta.

```
5 REM TEMPO DI RISPOSTA
10 PRINT "TEMPO DI RISPOSTA"
15 FOR I = 1 TO 20 * RND (100)
20 NEXT I
30 POKE 16414,0
40 POKE 16415,0
50 PRINT "SCHIACCIA (NEW LINE)"
60 INPUT C$
70 LET A = PEEK (16414)
80 LET B = PEEK (16415)
90 PRINT "TUO TEMPO DI RISPOSTA:"; (B * 256 + A − 4) * 20;
   "MILLISECONDI"
```

9. Programma per giocare a mordere il formaggio

(a) All'inizio vengono riempiti di 1 tre vettori A(10), B(10) e C(10); questi 3 vettori rappresentano le tre righe che vengono via via disegnate sul video. Nelle parti seguenti, viene analizzato ogni vettore e quindi ogni riga, e, se l'elemento è 1 viene stampato un quadratino nero (CHR$(128)), mentre se l'elemento è zero si ha uno spazio.
(b) Si analizza il vettore A e si stampa la prima riga.
(c) Si analizza il vettore B e si stampa la seconda riga.
(d) Si analizza il vettore C e si stampa la terza riga.
(e) Viene chiesto di premere NEW LINE per mordere il formaggio. Se si preme un tasto qualunque il programma va allo STOP. Se si preme NEW LINE si generano due numeri pseudo random $I \leqslant 10$ e $K \leqslant 3$; tali numeri sono le coordinate dell'elemento da azzerare in uno dei tre vettori. Si torna al ciclo di stampa in (b).

```
5 REM MORSI NEL FORMAGGIO
10 PRINT "MORSI NEL FORMAGGIO"
12 PRINT
15 DIM A(10)
20 DIM B(10)
```

```
30 DIM C(10)
100 FOR J = 1 TO 10
110 LET A(J) = 1
120 LET B(J) = 1
130 LET C(J) = 1
140 NEXT J
200 FOR J = 1 TO 10
205 IF NOT A(J) = 1 THEN GO TO 220
210 PRINT CHR$(128);
215 GO TO 230
220 PRINT " ";
230 NEXT J
240 PRINT
300 FOR J = 1 TO 10
305 IF NOT B(J) = 1 THEN GO TO 320
310 PRINT CHR$(128);
315 GO TO 300
320 PRINT " ";
330 NEXT J
340 PRINT
400 FOR J = 1 TO 10
405 IF NOT C(J) = 1 THEN GO TO 420
410 PRINT CHR$ (128);
415 GO TO 430
420 PRINT " ";
430 NEXT J
440 PRINT
442 PRINT
445 PRINT "PREMI (NEW LINE) PER MORDERE IL
    FORMAGGIO"
450 INPUT Y$
460 IF NOT Y$ = "" THEN GO TO 1000
470 CLS
500 LET I = RND(10)
510 LET K = RND(3)
520 IF K = 1 THEN LET A(I) = 0
530 IF K = 2 THEN LET B(I) = 0
540 IF K = 3 THEN LET C(I) = 0
550 GO TO 200
1000 STOP
```

10. Programma per provare il concatenamento dei FOR

Il solo scopo del programma è di far vedere che si possono concatenare 26 FOR, infatti le variabili di controllo diverse sono al massimo 26.

Salvo per il primo ciclo dove I varia da 1 a 2, per tutti gli altri FOR si ha un solo ciclo, cioè vengono percorsi una sola volta. Come output si ha la stampa di:

1 2

```
10 FOR A = 1 TO 2
20 FOR B = 1 TO 1
30 FOR C = 1 TO 1
40 FOR D = 1 TO 1
50 FOR E = 1 TO 1
60 FOR F = 1 TO 1
70 FOR G = 1 TO 1
80 FOR H = 1 TO 1
90 FOR I = 1 TO 1
100 FOR J = 1 TO 1
110 FOR K = 1 TO 1
120 FOR L = 1 TO 1
130 FOR M = 1 TO 1
140 FOR N = 1 TO 1
150 FOR O = 1 TO 1
160 FOR P = 1 TO 1
170 FOR Q = 1 TO 1
180 FOR R = 1 TO 1
190 FOR S = 1 TO 1
200 FOR T = 1 TO 1
210 FOR U = 1 TO 1
220 FOR V = 1 TO 1
230 FOR W = 1 TO 1
240 FOR X = 1 TO 1
250 FOR Y = 1 TO 1
260 FOR Z = 1 TO 1
300 PRINT A,
410 NEXT Z
420 NEXT Y
430 NEXT X
440 NEXT W
450 NEXT V
460 NEXT U
470 NEXT T
480 NEXT S
490 NEXT R
500 NEXT Q
510 NEXT P
520 NEXT O
530 NEXT N
540 NEXT M
550 NEXT L
560 NEXT K
```

```
570 NEXT J
580 NEXT I
590 NEXT H
600 NEXT G
610 NEXT F
620 NEXT E
630 NEXT D
640 NEXT C
650 NEXT B
660 NEXT A
```

11. Programma per provare ancora il concatenamento di 26 FOR

Con questo programma si ottengono ancora due cicli, per A = 1 e per A = 2, in ogni ciclo vengono stampati i valori delle variabili di controllo che governano i 25 cicli concatenati. Si ottiene in stampa:

```
1
2    3    4    5
6    7    8    9
10   11   12   13
14   15   16   17
18   19   20   21
22   23   24   25
26
2
2    3    4    5
6    7    8    9
10   11   12   13
14   15   16   17
18   19   20   21
22   23   24   25
26
```

```
10 FOR A = 1 TO 2
20 FOR B = 2 TO 2
30 FOR C = 3 TO 3
40 FOR D = 4 TO 4
50 FOR E = 5 TO 5
60 FOR F = 6 TO 6
70 FOR G = 7 TO 7
80 FOR H = 8 TO 8
90 FOR I = 9 TO 9
100 FOR J = 10 TO 10
110 FOR K = 11 TO 11
```

```
120 FOR L  = 12 TO 12
130 FOR M = 13 TO 13
140 FOR N  = 14 TO 14
150 FOR O  = 15 TO 15
160 FOR P  = 16 TO 16
170 FOR Q  = 17 TO 17
180 FOR R  = 18 TO 18
190 FOR S  = 19 TO 19
200 FOR T  = 20 TO 20
210 FOR U  = 21 TO 21
220 FOR V  = 22 TO 22
230 FOR W = 23 TO 23
240 FOR X  = 24 TO 24
250 FOR Y  = 25 TO 25
260 FOR Z  = 26 TO 26
300 PRINT A
305 PRINT
310 RINT B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,
    Q,R,S,T,U,V,W,X,Y,Z
315 PRINT
320 PRINT
410 NEXT Z
420 NEXT Y
430 NEXT X
440 NEXT W
450 NEXT V
460 NEXT U
470 NEXT T
480 NEXT S
490 NEXT R
500 NEXT Q
510 NEXT P
520 NEXT O
530 NEXT N
540 NEXT M
550 NEXT L
560 NEXT K
570 NEXT J
580 NEXT I
590 NEXT H
600 NEXT G
610 NEXT F
620 NEXT E
630 NEXT D
640 NEXT C
650 NEXT B
660 NEXT A
```

CAPITOLO 12

Il linguaggio macchina

Si è già detto all'inizio del libro che il linguaggio BASIC è un linguaggio simbolico ad alto livello, facile da apprendere e molto versatile, ma che per eseguire le frasi del linguaggio deve essere presente nella memoria del calcolatore un programma interprete, che traduca le frasi BASIC nel linguaggio proprio del calcolatore, quello che viene chiamato *il linguaggio macchina.*

Nel BASIC dello ZX80 si hanno 3 possibili punti di approccio al codice macchina; essi sono:

- l'istruzione POKE n, m che ci permette di scrivere nel byte di indirizzo n il valore m;
- la funzione PEEK(n), che ci permette di leggere il contenuto del byte di indirizzo n;
- la funzione USR(n) che ci permette di andare ad eseguire una sequenza di istruzioni in codice macchina a partire dall'indirizzo n. Questa funzione fornisce nei registri speciali HL dello ZX80 il valore calcolato, se a causa del calcolo il valore di HL è stato modificato, oppure ci fornisce il valore di n. Si può scrivere così: LET X = USR(n) e X verrà posto al valore contenuto in HL, oppure sarà X = n.

Gli appassionati possono studiare il codice macchina del microprocessore Z-80 e provare delle sequenze di programma in codice macchina sullo ZX80. Il programma in codice macchina di norma si scrive in notazione esadecimale e si ottiene una lista di valori da caricare a partire da un byte. Le istruzioni sono di lunghezza variabile e possono occupare uno o più bytes.

I contenuti dei bytes vanno convertiti in decimale e si possono facilmente caricare in memoria con un programma di questo tipo:

```
10 FOR I = n1 TO n2
20 PRINT I,
30 INPUT A
40 POKE I, A
50 PRINT PEEK(I)
60 NEXT I
```

dove n1 è bene sia scelto in modo che il programma in codice macchina si venga a trovare in memoria sopra il display file e prima dell'area STACK, n2 dipenderà dalla lunghezza del programma. Se per esempio si prende per n2 il valore 17000, poi nel programma BASIC si scriverà LET X = USR (17000) e si otterrà il valore calcolato.

Il programma di caricamento esposto ha lo svantaggio di non contenere le istruzioni del programma in linguaggio macchina, in quanto vengono ricevute come INPUT, questo significa che se si fa il SAVE del programma su nastro si perdono le istruzioni in linguaggio macchina. Con SAVE si memorizza solo il programma BASIC con i suoi dati. Si può allora scrivere per esteso una sequenza di caricamento, con una serie di POKE I, valore. In questo caso il programma BASIC ogni volta che viene usato carica la sequenza in linguaggio macchina, questa non va persa e viene memorizzata su nastro. Si ricordi comunque che la sequenza di istruzioni in linguaggio macchina deve terminare con una istruzione RET che si codifica $(C9)_{16}$ e quindi $(201)_{10}$, e serve a ritornare all'istruzione dopo la chiamata al sottoprogramma.

Segue un semplice esempio di programma in linguaggio macchina mandato in esecuzione, dopo averlo caricato in memoria, da un programma BASIC. Il programma chiede all'utente un numero e lo memorizza in N; stampa il numero che è stato dato come valore iniziale, va ad eseguire un programmino in linguaggio macchina che incrementa il numero di due unità, ritorna al programma BASIC che stampa il numero incrementato.

```
10 REM PROVA DI ISTRUZIONI IN LINGUAGGIO MACCHINA
20 REM SEQUENZA CARICAMENTO A PARTIRE DA BYTE 17000 DEL
30 REM PROGRAMMA IN LINGUAGGIO MACCHINA
40 POKE 17000,62
50 POKE 17001,00
60 POKE 17002,60
70 POKE 17003,60
80 POKE 17004,38
90 POKE 17005,00
100 POKE 17006,111
110 POKE 17007,201
115 REM CHIEDE IL NUMERO INIZIALE
120 PRINT "SCRIVI UN NUMERO N"
130 INPUT N
135 REM STAMPA IL VALORE INIZIALE DEL NUMERO
140 PRINT "VALORE INIZIALE N ="; N
145 REM SCRIVE NEL BYTE 17001 IL VALORE INIZIALE DI N
150 POKE 17001,N
155 REM VA AD ESEGUIRE ROUTINE IN LINGUAGGIO MACCHINA
160 X = USR (17000)
165 REM STAMPA IL VALORE CALCOLATO CHE SI TROVA IN X
170 PRINT "VALORE FINALE N ="; X
180 STOP
```

Il programma in linguaggio macchina caricato con le 8 istruzioni POKE nelle righe 40÷110 corrisponde alle seguenti istruzioni simboliche:

LDA, dato	istruzione	occupante	2	bytes,	nel secondo byte si pone inizialmente 00 carica nel registro A il secondo byte
INC A	”	”	1	”	incrementa il registro A di 1
INC A	”	”	1	”	”
LD H,00	”	”	2	”	carica nel registro H il secondo byte e quindi 00
LD L,A	”	”	1	”	carica nel registro L il registro A
RET	”	”	1	”	ritorna il controllo al programma che ha mandato alla routine

Come si vede il programma usa come accumulatore il registro A; terminato il calcolo trasferisce il valore nel registro L, ma azzera la parte H del registro doppio HL. Il programma lavora su N memorizzato nel byte 17001, quindi N deve essere minore o uguale a 253; si ricordi che l'indirizzo della variabile si riferisce al byte meno significativo. Un metodo utile per caricare programmi in linguaggio macchina può essere il seguente: è noto che il programma BASIC parte sempre dall'indirizzo 16424, si possono memorizzare i valori da caricare con la POKE in memoria in testa al programma con una REM, come numeri decimali di 3 cifre, poi in ciclo leggerli con una PEEK e scriverli con una POKE. Nel caso presentato sarebbe:

01 REM 06200006006003800011201.

Il numero di linea e la parola chiave REM occupano 3 bytes, per cui il primo valore inizia nel byte 16427 ed occupa 3 bytes. Il programma che carica la routine in linguaggio macchina e la stampa può essere il seguente:

```
10 REM PROGRAMMA CARICAMENTO
20 LET N = 16427      16427 indirizzo iniziale primo valore
30 LET M = 17000      17000 indirizzo iniziale per caricare il programma
40 LET X = PEEK(N)−28
50 LET Y = PEEK(N+1)−28
60 LET Z = PEEK(N+2)−28
70 LET X = X*100 + Y * 10 + Z
80 POKE M,X
90 PRINT M,X
100 IF X = 201 THEN GOTO 200
```

```
110 LET N = N + 3
120 LET M = M + 1
130 GOTO 40
200.....
```

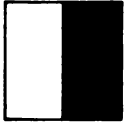
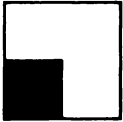
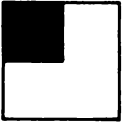
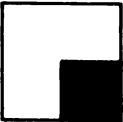
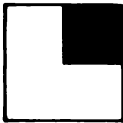
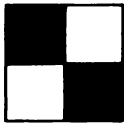
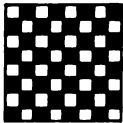
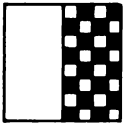
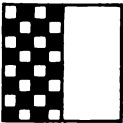
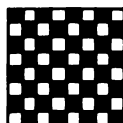
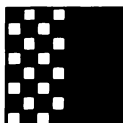
Nell'appendice C sono riportate le istruzioni in linguaggio macchina dello ZX80 ed alcune informazioni sui registri; si rimandano agli appassionati al libro "Il NANOBOOK Z-80 - vol. 1 - Tecniche di programmazione" pubblicato dal Gruppo Editoriale Jackson.

APPENDICE A

Caratteri del sistema

Riportiamo una tabella dei caratteri del sistema. Osservando questa tabella si vede che non tutti i caratteri sono disponibili sulla tastiera e che non tutti i codici possono essere stampati sullo schermo in modo diversificato. Si può provare a stampare i caratteri con un semplice programma di questo tipo:

10 INPUT X	si fornisce un codice numerico della tabella
15 IF X = 0 THEN GOTO 30	se X = 0 si salta allo STOP
20 PRINT CHR$(X)	si tenta di stampare il carattere il cui codice è X
25 GOTO 10	si torna a leggere un codice
30 STOP	si ferma il programma

Codice	Carattere	Codice	Carattere	Codice	Carattere
0	spazio	23	>	46	I
1	stringa nulla	24	<	47	J
2		25	;	48	K
3		26	,	49	L
4	da 2 a 11	27	.	50	M
5	caratteri	28	0	51	N
6	grafici	29	1	52	O
7	riportati	30	2	53	P
8	a parte	31	3	54	Q
9	a parte	32	4	55	R
10		33	5	56	S
11		34	6	57	T
12	£	35	7	58	U
13	$	36	8	59	V
14	:	37	9	60	W
15	?	38	A	61	X
16	(	39	B	62	Y
17	)	40	C	63	Z
18	−	41	D	64	?
19	+	42	E	da 65 a 127	?
20	*	43	F	128	spazio inverso
21	/	44	G	129	apici inversi
22	=	45	H		

Codice	Carattere	Codice	Carattere	Codice	Carattere
130		165	9 inverso	218	(
131	da 130 a	166	A ”	219	NOT
132	139	167	B ”	220	—
133	caratteri	168	C ”	221	+
134	grafici	169	D ”	222	*
135	in campo	170	E ”	223	/
136	inverso	171	F ”	224	AND
137	riportati a	172	G ”	225	OR
138		173	H ”	226	**
139		174	I ”	227	=
140	£ inverso	175	J ”	228	>
141	$ ”	176	K ”	229	<
142	: ”	177	L ”	230	LIST
143	? ”	178	M ”	231	RETURN
144	(”	179	N ”	232	CLS
145	) ”	180	O ”	233	DIM
146	- ”	181	P ”	234	SAVE
147	+ ”	182	Q ”	235	FOR
148	* ”	183	R ”	236	GOTO
149	/ ”	184	S ”	237	POKE
150	= ”	185	T ”	238	INPUT
151	> ”	186	U ”	239	RANDOMISE
152	< ”	187	V ”	240	LET
153	; ”	188	W ”	241	?
154	, ”	189	X ”	242	?
155	. ”	190	Y ”	243	NEXT
156	0 ”	191	Z ”	244	PRINT
157	1 ”	da 192 a 211	?	245	?
158	2 ”	212	”	246	NEW
159	3 ”	213	THEN	247	RUN
160	4 ”	214	TO	248	STOP
161	5 ”	215	;	249	CONTINUE
162	6 ”	216	,	250	IF
163	7 ”	217	)	251	GOSUB
164	8 ”			252	LOAD
				253	CLEAR
				254	REM
				255	?

Se con il programma che precede questa tabella si pone X = 212 si ottiene con CHR$(212) la stampa degli apici, altrimenti non ottenibili all'interno di una normale stringa.

Inverso significa che il carattere compare bianco in campo scuro.

L'inverso dello spazio è un quadrato scuro.

I caratteri grafici sono:

SIMBOLO CODICE
2
3
4
5
6
SIMBOLO CODICE
7
8
9
10
11
SIMBOLO CODICE
130
131
132
133
134
SIMBOLO CODICE
135
136
137
138
139

APPENDICE B

Variabili del sistema

Il contenuto dei primi 40 bytes della RAM è quello spiegato qui di seguito. Alcune variabili occupano un byte, altre due. Con le due istruzioni POKE e PEEK si possono scrivere e leggere queste variabili. Se le variabili sono di 1 byte non ci sono problemi. Se le variabili sono di 2 bytes per scrivere una variabile di valore V all'indirizzo n si deve procedere così:

```
POKE n*1,V/256          si scrive la parte intera di V/256.
POKE n, V-V/256*256     si scrive il resto della divisione precedente.
```

Analogamente per conoscere il valore V di una variabile di 2 bytes di indirizzo n, si deve procedere così: PEEK(n) + PEEK(n+1)*256, se si è sicuri che la variabile è positiva.
Se la variabile può essere negativa si deve invece procedere così:

```
LET MSB = PEEK (n+1)
IF MSB > 127 THEN LET MSB = MSB - 256
LET V = PEEK(n) + MSB * 256.
```

Nella tabella che segue valgono queste convenzioni:

X	significa che la variabile non può essere modificata, se lo si fa il sistema si blocca;
N	si può anche modificare la variabile dato che il sistema la rigenera;
1 o 2	a indicare se occupa 1 byte o 2 bytes;
U	a indicare variabile non segnata, cioè da 0 a 65535, il BASIC tratta questa variabile considerando i valori da 32768 a 65535 come valori da − 32768 a − 1.

Note	Indirizzi	Commenti
1	16384	contiene il numero dell'errore accaduto − 1, normalmente contiene 255. Se capita un errore di supero di capacità, codice 6, essa contiene 5. Se si ha una POKE per scriverci qualcosa si devono usare solo i numeri 255 oppure tra 0 e 8. Se si scrive POKE 16384,8 si ha STOP, infatti 8 + 1 = 9 codice dello STOP.
X1	16385	flag usati dal sistema, cioè indicatori interni.
2	16386	numero della linea in esecuzione. POKE non ha effetto a meno che non sia l'ultima linea del programma.
N2	16388	posizione in RAM (zona video) del cursore K o L dello schermo.
2	16390	numero della linea alla quale si trova il puntatore di linea.
X2	16392	VARS si veda cap. 10.
X2	16394	E-LINE si veda cap. 10.
X2	16396	D-FILE si veda cap. 10.
X2	16398	DF-EA si veda cap. 10
X2	16400	DF-END si veda cap. 10.
X1	16402	numero di linee della parte bassa dello schermo, inclusa la linea in bianco che separa le due parti.
2	16403	numero della linea che appare per prima sullo schermo. Viene modificato da LIST e quando lo schermo elimina le prime linee.
2	16405	indirizzo di quello che precede il cursore marcatore di errore S.
2	16407	numero della linea alla quale fa saltare CONTINUE.
N1	16409	flag usati dal sistema per controllare la sintassi delle frasi.
N2	16410	indirizzo del prossimo elemento nella tabella della sintassi.

Note	Indirizzi	Commenti
U2	16412	punto di partenza per il generatore dei numeri random. Viene modificato da RANDOMISE ed aggiornato ogni volta che si usa RND.
U2	16414	numero dei fotogrammi dello schermo dal momento dell'accensione dello ZX80. Più esattamente il resto quando questo è diviso per 65536. Quando si ha una immagine sullo schermo, questo contatore è incrementato 50 volte al secondo per la versione UK e 60 volte al secondo per la versione US.
N2	16416	indirizzo del primo carattere del nome della prima variabile in frasi LET, INPUT, FOR, NEXT, DIM.
N2	16418	valore dell'ultima espressione o variabile.
X1	16420	posizione nella linea attuale del prossimo carattere da scrivere sullo schermo, dove 33 significa ultima a sinistra, 32 la seconda da sinistra, 2 l'ultima a destra, 1 prima colonna nella prossima linea perché la linea attuale è piena, 0 prima colonna nella prossima linea perché è arrivato segnale di fine linea (dopo una PRINT che non termina con virgola o punto e virgola). Si ha il valore 33 se lo schermo è vuoto, per esempio dopo un CLS.
X1	16421	posizione della linea attuale sullo schermo; 23 significa linea in alto, 22 seconda linea, ecc.
X2	16422	indirizzo del carattere dopo la parentesi chiusa in PEEK oppure del NEW LINE alla fine della frase POKE.

NOTA: Si segnala l'algoritmo usato per generare i numeri pseudo-random. Sia n il valore contenuto in 16412; se n = 0 si pone n = 65536. Si calcola il resto m di (n * 77)/65537, se m = 65536 si pone m = 0. Il risultato di RND(x) è X * m/65536. m viene posto in 16412.

APPENDICE C

Scheda BASIC ZX80

Variabili:

Intere — primo carattere alfabetico, caratteri successivi o cifre o lettere senza spazi, contenuto numeri interi compresi tra −32768 e + 32767.

Stringhe: — nome formato da una lettera seguita da $ (dollaro), non c'è limite al numero dei caratteri contenuti.

Costanti

Intere: — numeri compresi fra − 32768 e + 32767.

Stringhe: — delimitate dagli apici, lunghezza a piacere, possono contenere qualunque carattere salvo gli apici.

Variabili con indice:

Intere: — nome formato da una sola lettera, un solo indice e come indice espressione intera.

Variabili di controllo:

Intere: — e con nome formato da una sola lettera.

Espressioni aritmetiche:

Operatori aritmetici:

** (elevato a)
- (unitario)
* prodotto
/ divisione
+ somma (somma e sottrazione non hanno ordine
− sottrazione di precedenza tra loro)
uso delle parentesi
ordine di valutazione da sinistra a destra con la precedenza con la quale sono stati listati gli operatori.

Espressioni relazionali:	*Operazioni relazionali*	= > < (senza ordine di precedenza tra loro) Valore − 1 per condizione vera; 0 per condizione falsa.
	Operatori logici:	NOT, AND, OR (le precedenze sono quelle date dalla lista)
Espressioni Booleane:	usano gli operatori logici.	
Istruzioni:	NEW	inizializza il calcolatore e cancella la memoria.
	LOAD	carica programmi e dati da nastro magnetico.
	SAVE	memorizza programmi e dati su nastro magnetico.
	RUN	manda in esecuzione un programma azzerando le variabili.
	RUN n	come sopra, ma con partenza dalla linea n.
	CONTINUE	fa continuare da n se n è nel messaggio del sistema, fa continuare da n + 1 dopo uno STOP.
	REM	commenti a scopo documentativo.
	IF n THEN istruzione	esegue istruzione se la condizione n è vera.
	INPUT dest	legge e memorizza in dest.
	PRINT lista	stampa il contenuto di lista, separatori di campo; e.
	LIST n	lista il programma con il puntatore di linea ad n.
	LIST	lista il programma dall'inizio.
	STOP	ferma il programma, per continuare CONTINUE.

DIM A(n)	predispone una variabile numerica con indice formata da n + 1 elementi.
FOR K = n1 TO n2	gestisce con il contatore K un ciclo per valore iniziale di K = n1 e valore finale di K = n2 dando ad ogni giro l'incremento di 1 a K.
GOTO n	salta alla linea n.
POKE n1, n2	scrive all'indirizzo n1 il valore n2.
RANDOMISE n	pone l'inizio per la generazione dei numeri a caso al valore n.
RANDOMISE	come sopra, ma n = valore del contatore dei fotogrammi dello schermo.
CLEAR	cancella tutte le variabili.
CLS	azzera la parte superiore dello schermo.
GOSUB n	come GOTO, ma conserva nello STACK l'indicazione per ritornare al programma principale.
RETURN	fa prelevare dallo STACK l'indicazione per tornare al programma principale.
NEXT K	chiude il ciclo iniziato da FOR, incrementa K e ne controlla il valore.
LET	consente di fare qualunque operazione di assegnazione o di calcolo.

Esiste anche il tasto BREAK per interrompere l'esecuzione di un programma se non è in attesa di INPUT.

Funzioni implementate: RND(n) genera un numero pseudo-random minore o uguale a n. La sequenza è influenzata nel punto di partenza da RANDOMISE.

ABS (espress.)	fornisce il valore assoluto dell'espressione.
PEEK(n)	fornisce il contenuto del byte di memoria di indirizzo n.
USR(n)	permette di andare ad eseguire un codice macchina memorizzato a partire da n.
CHR$(x)	fornisce il carattere corrispondente al codice numerico x
TL$ (stringa)	ritorna la stringa senza il suo primo carattere.
CODE (stringa)	fornisce il codice numerico del primo carattere della stringa.
STR$(x)	fornisce una stringa di caratteri corrispondente al numero x

APPENDICE D

Linguaggio macchina

Si riporta una tabella contenente le istruzioni in linguaggio macchina dello ZX80, la traduzione in esadecimale e decimale ed una breve spiegazione del significato di ogni istruzione. Si noti che i valori decimali vanno da 0 a 256 e quindi, ciò che, se si lavora in assoluto, viene interpretato come una istruzione in linguaggio macchina, se si lavora in BASIC ha un significato completamente diverso.

Codice operativo	Valore Esadec.	Valore Decimale	Commento
NOP	00	0	nessuna operazione
LD BC, nn	01	1	il numero nn è caricato in BC
LD (BC),A	02	2	il contenuto di A va nel byte puntato da BC
INC BC	03	3	incrementa BC di 1
INC B	04	4	incrementa B di 1
DEC B	05	5	decrementa B di 1
LD B,n	06	6	il numero n è caricato in B
RLCA	07	7	rotazione circolare sinistra dell'accumulatore
EX AF,AF'	08	8	scambia AF con AF'
ADD HL,BC	09	9	somma la contenuto di HL quello di BC
LD A,(BC)	0A	10	il contenuto del byte puntato da BC va in A
DEC BC	0B	11	decrementa BC di 1
INC C	0C	12	incrementa C di 1
DEC C	0D	13	decrementa C di 1
LD C,n	0E	14	il numero n è caricato in C
RRCA	0F	15	rotazione circolare destra dell'accumulatore
DJNZ disp	10	16	decrementa B e salta se B è diverso da 0
LD DE,nn	11	17	il numero nn è caricato in DE
LD (DE),A	12	18	il contenuto di A va nel byte puntato da DE
INC DE	13	19	incrementa DE di 1
INC D	14	20	incrementa D di 1

Codice operativo	Valore Esadec.	Valore Decimale	Commento
DEC D	15	21	decrementa D di 1
LD D,n	16	22	il numero n è caricato in D
RLA	17	23	rotazione sinistra dell'accumulatore
JR disp	18	24	salto relativo incondizionato
ADD HL,DE	19	25	somma al contenuto di HL quello di DE
LD A,(DE)	1A	26	il contenuto del byte puntato da DE va in A
DEC DE	1B	27	decrementa DE di 1
INC E	1C	28	incrementa E di 1
DEC E	1D	29	decrementa E di 1
LD E,n	1E	30	il numero n è caricato in E
RRA	1F	31	rotazione destra dell'accumulatore
JR NZ,disp	20	32	se Z = 1 continua, se Z = 0 PC = PC + DISP
LD HL,nn	21	33	il numero nn è caricato in HL
LD (nn),HL	22	34	H → (nn+1), L → (nn)
INC HL	23	35	incrementa HL di 1
INC H	24	36	incrementa H di 1
DEC H	25	37	decrementa H di 1
LD H,n	26	38	il numero n è caricato in H
DAA	27	39	converte in BCD il risultato
JR Z, disp	28	40	se Z = 0 continua, se Z = 1 PC = PC + DISP
ADD HL,HL	29	41	moltiplica per 2 il contenuto di HL
LD HL,(nn)	2A	42	il contenuto del byte (nn) va in HL
DEC HL	2B	43	decrementa HL di 1
INC L	2C	44	incrementa L di 1
DEC L	2D	45	decrementa L di 1
LD L,n	2E	46	il numero n è carcato in L
CPL	2F	47	complementa a 1 i bits di A
JR NC,disp	30	48	se C = 1 continua, se C = 0 PC = PC + DISP
LD SP,nn	31	49	il numero nn è caricato in SP
LD (nn),A	32	50	il contenuto di A va nel byte (nn)
INC SP	33	51	incrementa SP di 1
INC (HL)	34	52	incrementa di 1 il contenuto del byte (HL)
DEC (HL)	35	53	decrementa di 1 il contenuto del byte (HL)
LD (HL),n	36	54	il numero n è caricato nel byte (HL)
SCF	37	55	porre = 1 il flag di carry
JR C, disp	38	56	se C = 0 continua, se C = 1 PC = PC + DISP
ADD HL,SP	39	57	somma al contenuto di HL quello di SP

Codice operativo	Valore Esadec.	Valore Decimale	Commento
LD A,(nn)	3A	58	il contenuto del byte (nn) va in A
DEC SP	3B	59	decrementa SP di 1
INC A	3C	60	incrementa A di 1
DEC A	3D	61	decrementa A di 1
LD A,n	3E	62	il numero n è caricato in A
CCF	3F	63	complementa a 1 il flag di carry
LD B,B	40	64	carica B in B
LD B,C	41	65	carica C in B
LD B,D	42	66	carica D in B
LD B,E	43	67	carica E in B
LD B,H	44	68	carica H in B
LD B,L	45	69	carica L in B
LD B,(HL)	46	70	il contenuto del byte (HL) va in B
LD B,A	47	71	carica A in B
LD C,B	48	72	carica B in C
LD C,C	49	73	carica C in C
LD C,D	4A	74	carica D in C
LD C,E	4B	75	carica E in C
LD C,H	4C	76	carica H in C
LD C,L	4D	77	carica L in C
LD C,(HL)	4E	78	il contenuto del byte (HL) va in C
LD C,A	4F	79	carica A in C
LD D,B	50	80	carica B in D
LD D,C	51	81	carica C in D
LD D,D	52	82	carica D in D
LD D,E	53	83	carica E in D
LD D,H	54	84	carica H in D
LD D,L	55	85	carica L in D
LD D,(HL)	56	86	il contenuto del byte (HL) va in D
LD D,A	57	87	carica A in D
LD E,B	58	88	carica B in E
LD E,C	59	89	carica C in E
LD E,D	5A	90	carica D in E
LD E,E	5B	91	carica E in E
LD E,H	5C	92	carica H in E
LD E,L	5D	93	carica L in E
LD E,(HL)	5E	94	il contenuto del byte (HL) va in E
LD E,A	5F	95	carica A in E
LD H,B	60	96	carica B in H
LD H,C	61	97	carica C in H
LD H,D	62	98	carica D in H
LD H,E	63	99	carica E in H
LD H,H	64	100	carica H in H
LD H,L	65	101	carica L in H
LD H,(HL)	66	102	il contenuto del byte (HL) va in H

Codice operativo	Valore Esadec.	Valore Decimale	Commento
LD H,A	67	103	carica A in H
LD L,B	68	104	carica B in L
LD L,C	69	105	carica C in L
LD L,D	6A	106	carica D in L
LD L,E	6B	107	carica E in L
LD L,H	6C	108	carica H in L
LD L,L	6D	109	carica L in L
LD L,(HL)	6E	110	il contenuto del byte (HL) va in L
LD L,A	6F	111	carica A in L
LD (HL),B	70	112	carica B nel byte (HL)
LD (HL),C	71	113	carica C nel byte (HL)
LD (HL),D	72	114	carica D nel byte (HL)
LD (HL),E	73	115	carica E nel byte (HL)
LD (HL),H	74	116	carica H nel byte (HL)
LD (HL),L	75	117	carica L nel byte (HL)
HALT	76	118	HALT per la CPU
LD (HL),A	77	119	carica A nel byte (HL)
LD A,B	78	120	carica B in A
LD A,C	79	121	carica C in A
LD A,D	7A	122	carica D in A
LD A,E	7B	123	carica E in A
LD A,H	7C	124	carica H in A
LD A,L	7D	125	carica L in A
LD A,(HL)	7E	126	il contenuto del byte (HL) va in A
LD A,A	7F	127	carica A in A
ADD A,B	80	128	somma B ad A
ADD A,C	81	129	somma C ad A
ADD A,D	82	130	somma D ad A
ADD A,E	83	131	somma E ad A
ADD A,H	84	132	somma H ad A
ADD A,L	85	133	somma L ad A
ADD A,(HL)	86	134	somma (HL) ad A
ADD A,A	87	135	moltiplica per 2 il contenuto di A
ADC A,B	88	136	somma ad A, il contenuto di B + il Carry
ADC A,C	89	137	somma ad A il contenuto di C + il Carry
ADC A,D	8A	138	somma ad A il contenuto di D + il Carry
ADC A,E	8B	139	somma ad A il contenuto di E + il Carry
ADC A,H	8C	140	somma ad A il contenuto di H + il Carry
ADC A,L	8D	141	somma ad A il contenuto di L + il Carry
ADC A,(HL)	8E	142	somma ad A (HL) + il Carry
ADC A,A	8F	143	moltiplica x 2 A + il Carry

Codice operativo	Valore Esadec.	Valore Decimale	Commento
SUB B	90	144	sottrae ad A il contenuto di B
SUB C	91	145	sottrae ad A il contenuto di C
SUB D	92	146	sottrae ad A il contenuto di D
SUB E	93	147	sottrae ad A il contenuto di E
SUB H	94	148	sottrae ad A il contenuto di H
SUB L	95	149	sottrae ad A il contenuto di L
SUB (HL)	96	150	sottrae ad A il contenuto di (HL)
SUB A	97	151	sottrae ad A il contenuto di A
SBC A,B	98	152	A = A - B - Carry
SBC A,C	99	153	A = A - C - Carry
SBC A,D	9A	154	A = A - D - Carry
SBC A,E	9B	155	A = A - E - Carry
SBC A,H	9C	156	A = A - H - Carry
SBC A,L	9D	157	A = A - H - Carry
SBC A,(HL)	9E	158	A = A - (HL) - Carry
SBC A,A	9F	159	A = A - A - Carry
AND B	A0	160	A = AND logico fra A e B, mod. flags
AND C	A1	161	A = AND logico fra A e C, mod. flags
AND D	A2	162	A = AND logico fra A e D, mod. flags
AND E	A3	163	A = AND logico fra A e E, mod. flags
AND H	A4	164	A = AND logico fra A e H, mod. flags
AND L	A5	165	A = AND logico fra A e L, mod. flags
AND (HL)	A6	166	A = AND logico fra A ed il byte (HL)
AND A	A7	167	A = AND logico fra A e sè stesso
XOR B	A8	168	A = XOR fra A e B, modifica flags
XOR C	A9	169	A = XOR fra A e C, modifica flags
XOR D	AA	170	A = XOR fra A e D, modifica flags
XOR E	AB	171	A = XOR fra A e E, modifica flags
XOR H	AC	172	A = XOR fra A e H, modifica flags
XOR L	AD	173	A = XOR fra A e L, modifica flags
XOR (HL)	AE	174	A = XOR fra A e byte (HL), modifica flags
XOR A	AF	175	A = XOR fra A e sè stesso, modifica flags
OR B	B0	176	A = OR fra A e B, modifica frags
OR C	B1	177	A = OR fra A e C, modifica frags
OR D	B2	178	A = OR fra A e D, modifica frags
OR E	B3	179	A = OR fra A e E, modifica frags
OR H	B4	180	A = OR fra A e H, modifica frags
OR L	B5	181	esegue l'OR logico su L
OR (HL)	B6	182	esegue l'OR logico sul byte (HL)
OR A	B7	183	esegue l'OR logico su A
CP B	B8	184	sottrae B da A, modifica i flags
CP C	B9	185	sottrae C da A, modifica i flags
CP D	BA	186	sottrae D da A, modifica i flags
CP E	BB	187	sottrae E da A, modifica i flags

Codice operativo	Valore Esadec.	Valore Decimale	Commento
CP H	BC	188	sottrae H da A, modifica i flags
CP L	BD	189	sottrae L da A, modifica i flags
CP (HL)	BE	190	sottrae il byte (HL) da A, modifica i flags
CP (A)	BF	191	sottrae A da A, modifica i flags
RET NZ	C0	192	se la condizione è vera, return
POP BC	C1	193	BC è caricato con i 2 bytes dello stack
JP NZ,nn	C2	194	se la condizione è vera, salta a nn
JP nn	C3	195	salto incondizionato assoluto
CALL NZ,nn	C4	196	se la condizione è vera, CALL nn
PUSH BC	C5	197	salva BC in 2 bytes dello stack
ADD A,n	C6	198	somma il numero n ad A
RST 0	C7	199	accesso allo stack per salto a sottoprogramma
RET Z	C8	200	se la condizione è vera, return
RET	C9	201	return incondizionato
JP Z,nn	CA	202	se la condizione è vera, salta a nn
Istruz. a 2 byte	CB	203	prefisso per operazioni sui bits
CALL Z,nn	CC	204	se la condizione è vera, CALL nn
CALL nn	CD	205	chiamata subroutine
ADC A,n	CE	206	carica in A la somma A+A+Carry
RST 8	CF	207	accesso alla stack per salto a sottoprogramma
RET NC	D0	208	se la condizione è vera return
POP DE	D1	209	D ← (SP + 1), E ← (SP)
JP NC,nn	D2	210	se la condizione è vera PC = nn
OUT port,A	D3	211	A → port
CALL NC,nn	D4	212	se la condizione è vera CALL nn
PUSH DE	D5	213	(SP-2) ← E, (SP-1) ← D
SUB n	D6	214	sottrae il numero n ad A
$RST(10)_{16}$	D7	215	accesso allo stack per salto a sottoprogramma
RET C	D8	216	se la condizione è vera return
EXX	D9	217	scambia i due set di registri
JP C,nn	DA	218	se la condizione è vera PC = nn
IN A,port	DB	219	porta → A
CALL C,nn	DC	220	se la condizione è vera CALL nn
Istruz. a 2 byte	DD	221	indirizzamento indicizzato con ix+d
SBC A,n	DE	222	sottrae n ed il Carry da A
$RST\ (18)_{16}$	DF	223	accesso allo stack per salto a sottoprogramma
RET PO	E0	224	se la condizione è vera return
POP HL	E1	225	H ← (SP+1), L ← (SP)
JP PO,nn	E2	226	se la condizione è vera PC = nn

Codice operativo	Valore Esadec.	Valore Decimale	Commento
EX (SP),HL	E3	227	scambia H con (SP+1) e L con (SP)
CALL PO,nn	E4	228	se la condizione è vera CALL nn
PUSH HL	E5	229	(SP-2) ← L (SP-1) ← H
AND n	E6	230	esegue l'AND del numero n con A
RST $(20)_{16}$	E7	231	accesso allo stack per salto a sottoprogramma
RET PE	E8	232	se la condizione è vera return
JP (HL)	E9	233	PC = HL
JP PE,nn	EA	234	se la condizione è vera PC = nn
EX DE,HL	EB	235	scambia i contenuti tra DE e HL
CALL PE,nn	EC	236	se PE = 0 continua, se PE = 1 CALL nn
Istruz. a 2 byte	ED	237	prefisso per usi diversi
XOR n	EE	238	OR esclusivo del numero n con A
RST $(28)_{16}$	EF	239	accesso allo stack per salto a sottoprogramma
RET P	F0	240	se P = 0 continua, se P = 1 ritorna
POP AF	F1	241	A ← (SP+), F ← (SP)
JP P,nn	F2	242	se P = 0 continua, se P = 1 PC = nn
DI	F3	243	0 → IFF
CALL P,nn	F4	244	se P = 0 continua, se P = 1 CALL nn
PUSH AF	F5	245	(SP-2) ← F, (SP-1) ← A
OR n	F6	246	OR del numero n con l'accumulatore
RST $(30)_{16}$	F7	247	accesso allo stack per salto a sottoprogramma
RET M	F8	248	se M = 0 continua, se M = 1 ritorna
LD SP,HL	F9	249	carica HL in SP
JP M,nn	FA	250	se M = 0 continua, se M = 1 PC = nn
EI	FB	251	1 → IFF
CALL M,nn	FC	252	se M = 0 continua, se M = 1 CALL nn
Istruz. a 2 byte	FD	253	indirizzamento indicizzato con iy+d
CP n	FE	254	sottrae al contenuto di A il valore n
RST $(38)_{16}$	FF	255	accesso allo stack per salto a sottoprogramma.

Caratteristiche

Registri generali con possibilità di utilizzo a coppie:

Principali		*Alternativi* (memorie tampone)	
Accumulatore	Flag	Accumulatore	Flag
A	F	A’	F’
Utilità Generale			
B	C	B’	C’
D	E	D’	E’
H	L	H’	L’
Utilizzi Speciali			
I	R		
(interrupt)	(refresh)		
IX	(index doppia lunghezza)		
IY	(index doppia lunghezza)		
SP	(stack pointer)		
PC	(program counter)		

Indice analitico

L. 4.500 Cod. 317B

La d.ssa Rita Bonelli, laureatasi in Matematica e Fisica presso l'Università di Milano, ha cominciato ad occuparsi di analisi e programmazione sin dai primordi.
Dopo aver per molti anni consolidato la sua esperienza come sistemista EDP su sistemi grandi e medi (di tutte le generazioni), attualmente si occupa anche di "personal".
Da 12 anni ha affiancato alla sua attività di consulente l'insegnamento, è titolare di una cattedra di informatica presso l'Istituto Industriale Statale Feltrinelli di Milano.

30

IMPARIAMO A PROGRAMMARE IN BASIC CON LO ZX80

R. Bonelli

GRUPPO EDITORIALE JACKSON